Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

РАЗРАБОТКА МУЛЬТИМЕДИЙНОГО ПРОДУКТА В ЖАНРЕ «ЯПОНСКАЯ РОЛЕВАЯ ПОШАГОВАЯ ИГРА»

Выпускная квалификационная работа
по направлению подготовки 09.03.02 Информационные системы и технологии
профилю подготовки «Информационные технологии в медиаиндустрии»

Идентификационный номер ВКР: 207

Екатеринбург 2019

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

К ЗАЩИТЕ ДОПУСКАЮ
Заведующий кафедрой ИС
_____ И. А. Суслова
«____» _____ 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА МУЛЬТИМЕДИЙНОГО ПРОДУКТА В ЖАНРЕ «ЯПОНСКАЯ РОЛЕВАЯ ПОШАГОВАЯ ИГРА»

Исполнитель:
обучающийся группы ИТм-403 А. А. Егармин
Руководитель:
канд. пед. наук, доцент Т. В. Чернякова
Нормоконтролер:
ст. преподаватель каф. ИС Н. В. Хохлова

Екатеринбург 2019

АННОТАЦИЯ

Выпускная квалификационная работа состоит из мультимедийного продукта в жанре японской ролевой пошаговой игры и пояснительной записки на 62 страницах, содержащей 45 рисунков, 4 таблицы, 30 источников литературы, а также 1 приложения на 2 страницах.

Ключевые слова: UNITY ENGINE, ИГРОВОЙ ДВИЖОК, ВИДЕОИГРА, АРХИТЕКТУРА КОМПЬЮТЕРНОЙ ИГРЫ, ЯПОНСКИЕ КОМПЬЮТЕРНЫЕ ИГРЫ

Егармин А. А., Разработка мультимедийного продукта в жанре «японская ролевая пошаговая игра»: выпускная квалификационная работа / А. А. Егармин; Рос. гос. проф.-пед. ун-т, Ин-т инж.-пед. образования, Каф. информ. систем и технологий. — Екатеринбург, 2019. — 62 с.

В работе описана разработка мультимедийного продукта и его реализация на движке Unity Engine.

Цель работы — разработка технической демонстрационной версии мультимедийного продукта в жанре «японская ролевая пошаговая игра». Для реализации цели были проанализированы существующие разработки, изучено необходимое программное обеспечение и технологии создания игр. Создана техническая демонстрационная версия мультимедийного продукта в жанре «японская ролевая пошаговая игра», состоящая из пяти уровней, с помощью игрового движка Unity Engine.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последнее время компьютерные игры приобретают невероятную популярность во всём мире. Во все времена существования человечества, игры являлись неотъемлемой составляющей его жизни. Кроме того, игры привносят разнообразие в рутинную повседневность.

Рынок видеоигр является самым большим сегментом мирового рынка цифрового контента. Прогнозы различных аналитических компаний говорят о дальнейшем стремительном развитии рынка в ближайшее время. Согласно прогнозам, объем мирового рынка видеоигр к 2021 году составит почти 130 миллиардов долларов, держа средние ежегодные темпы роста в период 2016–2021 годов на уровне 5,4 %.

Несмотря на смещение рынка видеоигр в сторону мобильных приложений, компьютерные игры не могут потерять свою популярность, так как именно эта аудитория является ядром всей игровой аудитории.

Как правило, разработкой видеоигр занимается компания, которая может быть представлена как одним человеком, так и командой разработчиков. Крупные коммерческие проекты, обычно, создаются командами разработчиков в пределах одной определенной компании, основной источник дохода которой заключается в производстве видеоигр для компьютеров или консолей. Создание проекта, в большинстве случаев, финансирует более крупная компания, являющаяся издателем, которая, когда разработка будет завершена, будет заниматься изданием игры и связанными с этим, тратами. Реже издатель сам может включать в себя несколько команд разработчиков, или же компания, являющаяся разработчиком, может разрабатывать проекты за свой счет и распространять их без участия сторонних компаний, с помощью средств цифровой дистрибуции.

Создание самых высокобюджетных видеоигр («ААА-проекты»), как правило, стоит десятки миллионов долларов и, в последнее временя, эта сум-

ма непрерывно росла, как и число команд разработчиков, вместе с временными рамками разработки. Средний бюджет таких проектов, которые выпускаются большими издателями и продаются на физических носителях — варьируется от 13 до 21 млн. долларов.

Благодаря тому, что рынок видеоигр развивается, многие компании разработчиков получили возможность разрабатывать свои проекты без финансовых и юридических обязательств перед издателями. Инди-игры (Indie games, от англ. independent video games — «независимые видео игры») — это такие видеоигры, которые были созданы сольными разработчиками или малыми компаниями без поддержки компании-издателя. Распространение подобных проектов происходит с помощью сервисов цифровой дистрибуции. Количество явлений, которые связаны с такими играми, в последнее время, ощутимо растет, в основном благодаря тому, что новых способов онлайн-дистрибуции и средств разработки продолжают развиваться.

Данный проект относится к инди-разработке и развивается только за счет средств его разработчика. Благодаря развитию программного обеспечения для разработки видеоигр, разработчику не потребуется тратить время на разработку игрового движка, что позволит сразу же приступить к работе непосредственно над проектом и значительно сократит время разработки.

Объектом работы являются японские ролевые пошаговые игры.

Предметом работы является техническая демонстрационная версия мультимедийного продукта и его разработка.

Целью данной работы является разработка технической демонстрационной версии мультимедийного продукта в жанре «японская ролевая пошаговая игра».

Техническая демонстрационная версия (англ. tech demo) — прототип, неполная версия продукта, которая создана с целью показать идею, производительность, метод или какие-либо особенности программного продукта. Технические демонстрационные версии могут использоваться как презента-

ции для партнёров, журналистов или потенциальных потребителей для того, чтобы убедить их в жизнеспособности данного продукта.

Такие версии нельзя путать с демо-сценами, которые, хоть и показывают новые программные методики, на самом деле являются отдельной формой компьютерного творчества. Техническая демонстрационная версия отображает не только внешние свойства проекта, но и его внутреннюю архитектуру, механизмы взаимодействия.

В соответствии с целью были определены задачи:

1. Изучить литературу и интернет-источники по темам, связанным с жанром японских ролевых игр, а также методами разработки мультимедиа продуктов (видеоигр).

2. Проанализировать существующие мультимедиа продукты в жанре японских ролевых пошаговых игр, выявить достоинства и недостатки.

3. Сформулировать требования к разрабатываемому мультимедиа продукту и с их учетом спроектировать дизайн.

4. Разработать архитектуру мультимедийного продукта с помощью игрового движка Unity Engine.

5. Реализовать и протестировать мультимедиа продукт при помощи игрового движка Unity Engine.

1 ЯПОНСКИЕ РОЛЕВЫЕ ПОШАГОВЫЕ ИГРЫ И МЕТОДЫ ИХ РАЗРАБОТКИ

1.1 Терминология и основные особенности жанра «японская ролевая пошаговая игра»

Компьютерные игры описываются не только сюжетом, техническими характеристиками, на которые обращает внимание пользователь, но и ряд других аспектов, позволяющих разработчикам анализировать, что может быть актуально у пользователей. Анализируя мир компьютерных игр, а также проводя исследования разных направлений следует рассмотреть ряд основных игровых понятий.

Основополагающее понятие игровой индустрии — компьютерная игра — это компьютерная программа, служащая для организации игрового процесса (геймплея), связи с партнёрами по игре, или сама выступающая в качестве партнёра.

Компьютерные игры различаются по жанрам, где жанр — общее понятие, которое отражает наиболее существенные свойства и связи явлений мира искусства, симбиоз формальных и содержательных особенностей произведения. В рассматриваемом случае в качестве произведения выступает компьютерная игра.

Одним из популярных жанров компьютерных игр является rogue-like игра, которая является поджанром компьютерных ролевых игр. Характерными особенностями классического rogue-like являются случайным образом генерируемые уровни, пошаговость и необратимость смерти персонажа — в случае его гибели игрок не сможет загрузить игру и должен начать её заново.

RPG, Role-Playing Games — жанр компьютерных игр, который основан на элементах игрового процесса настольных ролевых игр. В такой игре игрок управляет одним или несколькими персонажами, каждый из которых описан

набором характеристик, списком способностей и умений; примерами таких характеристик могут быть, показатели силы, ловкости, интеллекта, защиты, уклонения, уровень развития того или иного навыка и т. п.

JRPG, Japanese Role-Playing Games — поджанр, выделяемый среди компьютерных ролевых игр. Игры этого жанра имеют схожие, узнаваемые особенности стиля, подачи сюжета и игровой механики, разительно отличающие их от RPG «западного образца».

Главный герой — главный герой, центральное действующее лицо, актёр, играющий главную роль в произведении и т. д. Противопоставляется антагонисту.

Поединки в настоящем времени — поединки без остановки времени игры, для свершения игроком определенного действия.

Single player — в компьютерных играх вид игры, в которой принимает участие один человек.

Пошаговые бои — поединки с неограниченным временем на выбор атаки или действия.

Игровой искусственный интеллект — набор программных методик, которые используются в компьютерных играх для создания иллюзии интеллекта в поведении персонажей, управляемых компьютером.

Как и в японской анимации, в их играх необходимо гораздо больше, чем принято в западных играх, уделять внимания к дизайну персонажей, разработке сценария игры, проработке разных мелочей — именно это может привлечь покупателей к новым играм. Естественно, это влияет и на процентное соотношение жанров в играх.

Одним из переломных моментов в развитии компьютерной графики на территории Японии было появление одной из первых трехмерных игр «Virtual Fighter».

В двухмерных играх угол камеры фиксирован, в то время как в трехмерных играх используются различные положения камеры и это не могло не повлиять на визуальный ряд и возможности взаимодействия с виртуальной

средой. В наше время, качество компьютерной графики постоянно улучшается и это приводит к тому, что качество визуального ряда, ранее доступное только в виде заранее снятых сцен, служивших одним из наиболее привлекательных аспектов японских игр на мировом рынке, сейчас доступно непосредственно в геймплее.

Идея создания игр рассматриваемого жанра пришла в Японию из запада, однако японцы добавили много своего — сложные игровые миры, заметное влияние мифологии и характерной японцам философии, большая роль для хода игры взаимоотношений героев, вариативность конечного результата в зависимости от действий игрока, профессиональный дизайн персонажей.

Например, в создании одной из таких игр, «Chrono Trigger» — одной из лучших RPG-игр для SNES (Super Nintendo Entertainment System), принимал участие знаменитый Тория́ма Акира, и в последнее время не он один из числа мэтров манги и аниме принимает участие в проектах такого рода.

Началом жанра принято рассматривать игру 1984 года, разработанную Nihon Falcom для систем, разработанных Nippon Electric, и название данному проекту было — Dragon Slayer (рисунок 1).

Рисунок 1 — Dragon Slayer

Невзирая на удивительную несложность игровых компонентов, игра обрела неописуемый успех на территории Японии.

Dragon Slayer был одним из первых представителей игр в жанре ARPG (Action RPG), в котором он заложил установленные основные принципы. Пусть эта игра и непосредственно базировалась в отдельных прототипах ARPG составляющих игры, произведенной годом ранее этой же фирмой именно её принято рассматривать родоначальником жанра.

Игра рассказывала о путешествии по многоярусному подземелью, в исследовании которого игроку могли помочь такие внутриигровые способности, как карта и инвентарь, в котором могло быть ограниченное число объектов в количестве одной вещи. В отличие от разработанных ранее rogue-like игр, в которых изучение мира и битвы выполнялись в пошаговом порядке, в Dragon Slayer был сформирован акцент на экшн. Кроме того, в игре существовали головоломки, решение которых содействовало прохождению игровых зон.

Самыми явными последователями идей Dragon Slayer можно рассматривать такие серии игр, как Ys и The Legend of Zelda, в которых очень похожим способом основывается ключевой геймплей — поединки в реальном времени, головоломки, менеджмент предметов и многое другое.

О таком явлении, как японская ролевая игра, запад узнал только много лет спустя, когда в августе 1989 года в Америке появилась игра 86 года выпуска — Dragon Quest.

Ранние игры серий Dragon Quest и Final Fantasy в значительной степени схожи между собой в плане геймплея — герои передвигаются по мировой карте между локациями, сражаются в пошаговом режиме, повышают уровни, зарабатывают деньги, покупают снаряжение и дополнительные объекты.

JRPG — крайне многообразный жанр, две игры, представляющие по сущности JRPG, вполне могут сильно различаться друг от друга. Например, если сравнивать серии Tales of и Final fantasy, можно найти очень много отличительных черт, начиная с боевой системы и заканчивая основой сюжета.

Тем не менее, возможно отметить целый ряд качеств, свойственный многим видеоиграм жанра JRPG.

Главный герой. Как правило, основной персонаж спасает галактику, планету либо городок от скрытой, либо очевидной опасности. Персонаж может являться кем угодно, главное, чтобы его личность отработана вплоть до мельчайших элементов.

Персонажи. Второстепенные и эпизодические герои имеют все шансы быть проработаны крайне плохо, однако товарищи основного героя, а также ключевые противники, как правило, запоминаются на долгое время.

Игровой мир. Безупречной площадкой для игр этого жанра становятся фантастические царства, реже — просторы галактики и отдаленных планет, ещё реже — мир, в общих чертах напоминающий современность.

Игровой процесс. В JRPG можно выявить множество условностей: с одной стороны, это может быть жестко ограниченная интерактивность окружения, с другой — возможность принять участие в большом количестве мини-игр.

Сюжет. Персонаж или партия имеют все шансы заниматься чем угодно, однако в конечном счете обнаружится, что они избавляют мир от античного проклятья, сумасшедшего тирана, вторжения соседнего государства.

Боевая система. Изучение игрового мира происходит в настоящем времени, однако поединки могут являться пошаговыми. Для ранних JRPG типична пошаговая боевая система, в которой противники в порядке очереди обмениваются ударами, при этом перемещение бойцов по полю битвы невозможно. Для подобных боев героев и чудовищ перемещают в особую арену. Последующий период развития боевой концепции JRPG — active time battle (ATB), суть которой сводится к тому, что герой способен нападать не ранее, чем заполнится соответствующая шкала действия персонажа. В современных JRPG можно повстречать поединки в настоящем времени.

1.2 Основные представители жанра «японская ролевая пошаговая игра»

Шесть долгих лет привнесли в игровую японскую игровую индустрию колоссальное число идей, какие многим знакомы только по играм известной нам современности. Однако, тот факт, то, что данные идеи зародились в столь далёком прошлом, действительно потрясает воображение. В том числе и в эту эпоху заимствования мыслей, общество всё точно также устремляется поразить друг друга глубиной своей фантазии и изобретательности.

В качестве первого примера современных игр рассматриваемого жанра, можно привести игру Persona 5 от компании Atlus.

Данная игра хорошо симулирует жизнь города, сменяя времена года, погоду, время суток и разделяя дни на активности. Часть из них прописана заранее — это учеба и обязательные сюжетные события. Часть — определяется самим игроком. Каждое действие имеет свою стоимость времени, а когда день заканчивается — остается только идти спать.

В процессе продвижения по сюжету главный герой получает доступ в новые районы города, каждый из которых имеет свое функциональное предназначение.

Persona 5 продолжает развивать пошаговую боевую систему серии Shin Megami Tensei, отличающуюся от стандартных игр жанра. Авторы постарались максимально скрыть движение героев на отдельный экран, что достигается работой камеры, незаметными подгрузками арены и перерисованным меню. В результате игра, словно музыкальный клип с грамотным монтажом, красиво и незаметно для глаз переходит к битвам (рисунок 2).

Это образцовая сюжетная JRPG с безумным дизайном, необычной для видеоигр музыкой и невероятно насыщенной историей. В ней находится огромное количество контента и новых механик, которые открываются по мере прохождения, уникальные подземелья и отличный ритм [1].

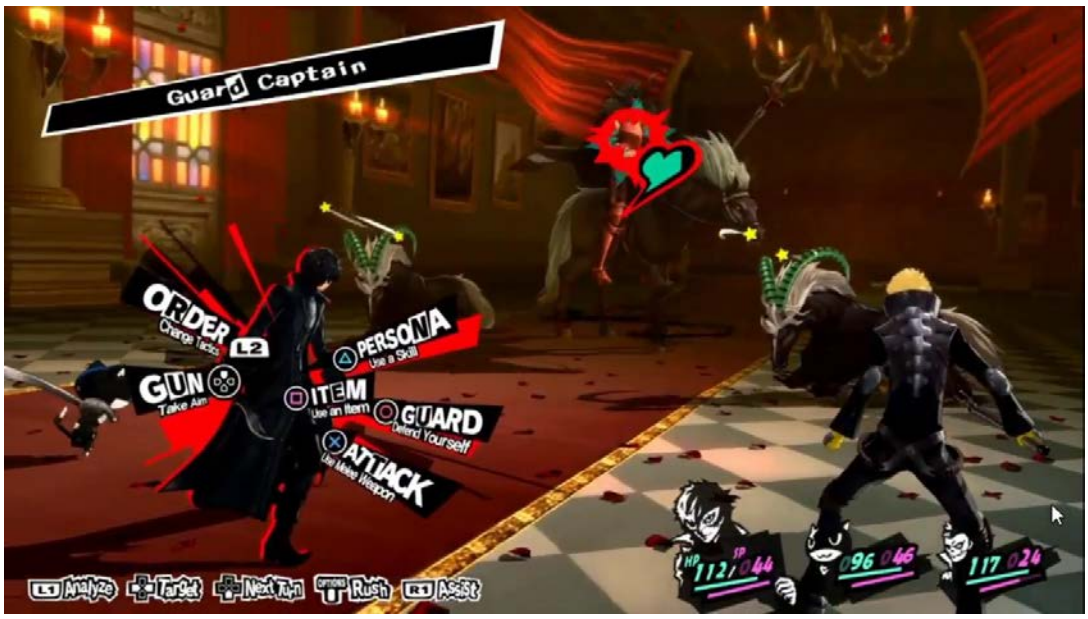

Рисунок 2 — Боевая система Persona 5

Вторым рассматриваемым примером является Tales of Berseria — шестнадцатая часть серии «Сказаний».

Сражения в Tales of Berseria происходят в реальном времени на небольшом поле боя, в пределах которого можно свободно перемещаться, атаковать или уклоняться.

Механика сражений основана на «шкале душ», которая представляет собой очки действий. Они тратятся на атаки и восстанавливаются во время бездействия. Как только их запас будет исчерпан, урон героя резко падает, а сам он раскроется, становясь уязвимым. Кроме обычных приемов у героев есть умение «Раскол души»: с его помощью они наносят противникам большой урон своими особыми способностями, за счет траты некоторого количества очков душ (рисунок 3).

Кроме стандартных сражений в Tales of Berseria есть несколько механик, которые делают игру более интересной. Например, в определенный момент игрок получит в своё распоряжение корабль, который можно отправлять на определенные задания. Мир игры — огромный архипелаг с островами в широком спектре климатических зон: от заснеженной тундры до тропического рая. И игроку дают возможность побывать почти на каждом из них, рассказывая потрясающую историю [1].

Рисунок 3 — Боевая система в Tales of Berseria

Третьим примером можно привести такую игру, как Final Fantasy XV (FF XV).

Сражения с монстрами — основная задача героев игры, так как именно эту планку серия задавала несколько десятилетий. Боевая система в FF XV является экшен-ориентированной. В ней, практически все завязано на атаке и парировании. Успешное парирование даст игроку возможность эффектно контратаковать и оглушить врага.

У главного героя появилась чрезвычайно полезная способность телепортироваться в любую часть боевой зоны. Ее можно использовать как для стратегического отступления, так и для атаки: герой запускает оружие во врага и молниеносно телепортируется в его сторону, чтобы нанести сильный удар.

Во время боя управлять можно только главным героем, все остальные персонажи подчинены искусственному интеллекту. Однако игрок волен сам применять специальную атаку союзника, которая становится возможной при заполнении «шкалы приема». Разработчики полностью отказались от возможности программировать поведение команды в бою. На замену стандарт-

ной механике пришли совместные атаки, которые срабатывают в зависимости от «контекста» (рисунок 4).

Также главный герой обладает уникальной магией: найденное в гробницах старых королей оружие объединяется воедино и наносит множество сильных ударов по врагу. Чем больше оружия игрок соберет в течение игры, тем мощнее будет атака [1].

Рисунок 4 — Бой в Final Fantasy XV

Чтобы не затеряться среди большого числа конкурентов и разработать интересный для потребителя продукт необходимо провести сравнительный анализ игр жанра, в котором ведется создание продукта. В таблице 1 можно рассмотреть сравнение различных игр рассматриваемого жанра [19].

Таблица 1 — Сравнение игр жанра Japanese Role-Playing Games разного времени

№	Название	Год выпуска	Мировая оценка Metacritic
1	Dragon Slayer	1984	8,6/10
2	Final fantasy	1987	8,9/10
3	Tales of Phantasia	1995	8,1/10
4	Chrono trigger	1995	9,2/10
5	Final Fantasy 7	1997	9,2/10
6	Final Fantasy 9	2000	9,3/10
7	Chrono Cross	2000	8,8/10
8	Final Fantasy 15	2016	7,7/10
9	Tales of Berseria	2017	5,5/10
10	Persona 5	2017	8,7/10

Несмотря на существенную разницу с западными, часто именно азиатские игры становятся передовыми в своей сфере, привносят новые элементы в жанр и надолго остаются в сердце игрока благодаря захватывающему сюжету и уникальному дизайну.

1.3 Анализ средств разработки ролевых пошаговых игр

В настоящее время, каждый желающий может самостоятельно начать разрабатывать игры используя разнообразные игровые движки, которыми пользуются не только начинающие или инди-разработчики, но и крупные всемирно известные компании.

Игровой движок — это программная среда, предназначенная для создания и разработки видеоигр или других интерактивных приложений с графикой, обрабатываемой в реальном времени. Движок игры включает в себя визуализатор, физический движок, систему скриптов, звук, анимацию, сетевой код, многопоточность и управление памятью.

Сначала каждая игра создавалась с нуля, но в середине 1990-х годов, после появления знаменитого шутера Doom, произошло сильное изменение концепции создания игр. Doom имел четкую архитектуру, которая легко делилась на центральные компоненты (передвижение, стрельба, обработчик столкновений и т.д.) и графические ресурсы, которые образовывали игровой мир. На основе подобной архитектуры стали создаваться игровые движки, после чего компании разработчиков могли использовать один и тот же набор инструментария для различных игр, что сокращало время и средства для разработки игры. В следствии этого, в последнее время, игры создаются либо на общедоступных движках, либо студии сами создают движки на основе ранее выпущенных ими игр, дорабатывая их, в последствии, под тенденции технического прогресса.

Сравнительная характеристика наиболее популярных движков представлена в таблице 2. Движки сравнивались по определенным характеристи-

кам, очень важным как для начинающего разработчика, так и для того, кто давно работает в данной сфере. Для разрабатываемого в рамках дипломного проекта, самыми важными факторами были — бесплатность распространения и удобство языка программирования.

Таблица 2 — Сравнительная характеристика игровых движков

Название	**Платформы**	**Язык программирования**	**Исходный код**	**Лицензия**	**Общие сведения**
Unreal Engine 4	Кроссплатформенный	C++, встроенная визуальная скриптовая система - Blueprint	Открытый	Free, если доходы от игры не превышают 3000$	Высокие графические возможности. Имеет мощный инструмент для дизайна игровых уровней прямо в движке.
Unity3D	Кроссплатформенный	C#, Java script	Закрытый	Personal-версия: free Premium-версия: 1500$	Хорошее качество графики, не требующая при создании игр больших познаний в языках программирования.
Havok Physics 8	Кроссплатформенный физический движок	C++	Закрытый	Комерческий	Симулирует физическое взаимодействия в реальном времени,
Creation Kit	Кроссплатформенный	Специальный язык Papyrus	Открытый	Free	Применяется в основном для создания модификаций к ролевой игре Skyrim

Для разработки однопользовательской игры в жанре JRPG был выбран движок Unity Engine, так как он имеет интуитивно-понятный интерфейс, удобную скриптовую систему на языке C#, который изучался во время обучения в университете, является наиболее инновационным и имеет большое сообщество пользователей. В документации описан инструментарий движка,

имеется справочник по работе с C# и ответы на часто задаваемые вопросы, например, пошаговая инструкция, как реализовать движение персонажа [4].

Для анимации моделей, был выбран сайт-сервис mixamo.com. Для того, чтобы начать работу с моделью, необходимо загрузить её в формате *.obj или *.fbx. Для работы с сайтом необходимо иметь Adobe ID, чтобы авторизоваться на сайте. Здесь для модели автоматически присваивается скелет, после чего её можно просто скачать в формате *.fbx либо добавить модели анимацию и скачать её в этом же формате уже с анимацией (рисунок 5).

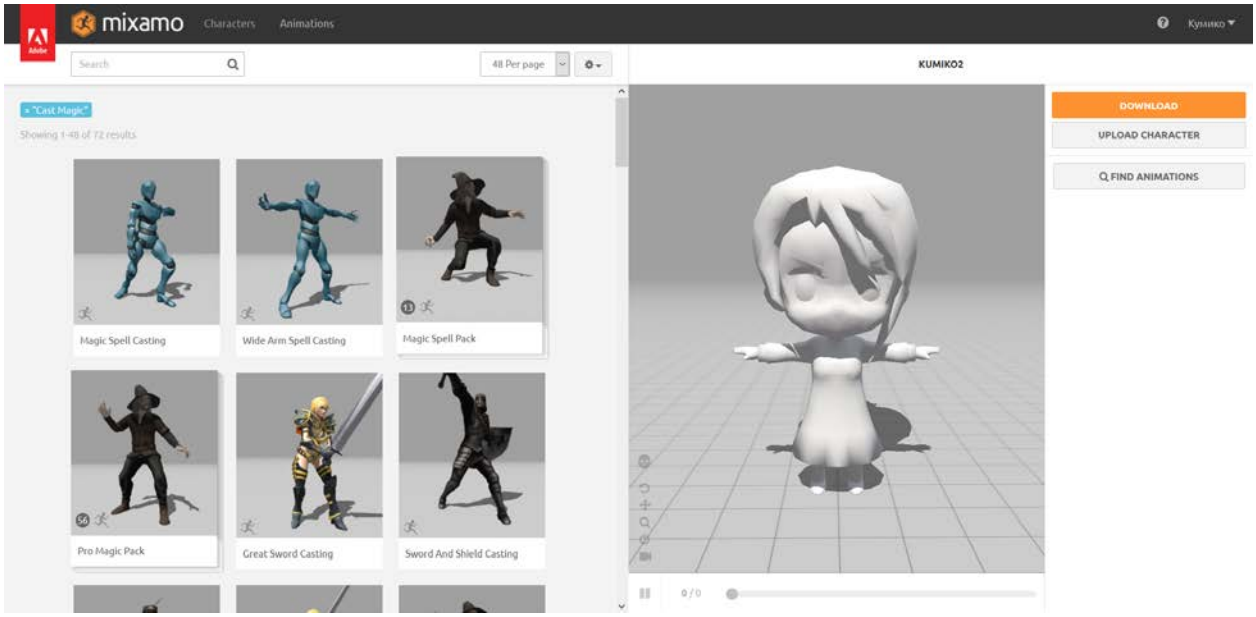

Рисунок 5 — Интерфейс сервиса Mixamo

Для того, чтобы сократить работу над графической составляющей игры и уделить больше внимания технической, было решено следующее: модели, найденные в Интернете или официальном Unity Asset Store, загружались на сайт, где происходил подбор анимаций, которые будут необходимы для реализации игры — бег, ходьба, состояние покоя и прочее.

Для разработки эскиза локации была выбрана программа RPG Maker MV (рисунок 6), так как она позволяет рассмотреть создаваемый проект с другой точки зрения. Данная программа ориентирована на создание 2D игр жанра JRPG и имеет множество настроек, связанных с настройкой локаций — например, система слоев, которая отвечает за какие спрайты игрок может пройти, а за какие нет. Также, в данной программе можно рассмотреть, какие

предметы и как лучше расставить на локации, чтобы это смотрелось логично. Также, с помощью данной программы имеется возможность создания референсов для персонажей, которые будут участвовать в игре, благодаря такому инструменту как «Генератор персонажей» (рисунок 7). Он предлагает множество различных возможностей для кастомизации героя, делая его, практически уникальным [9].

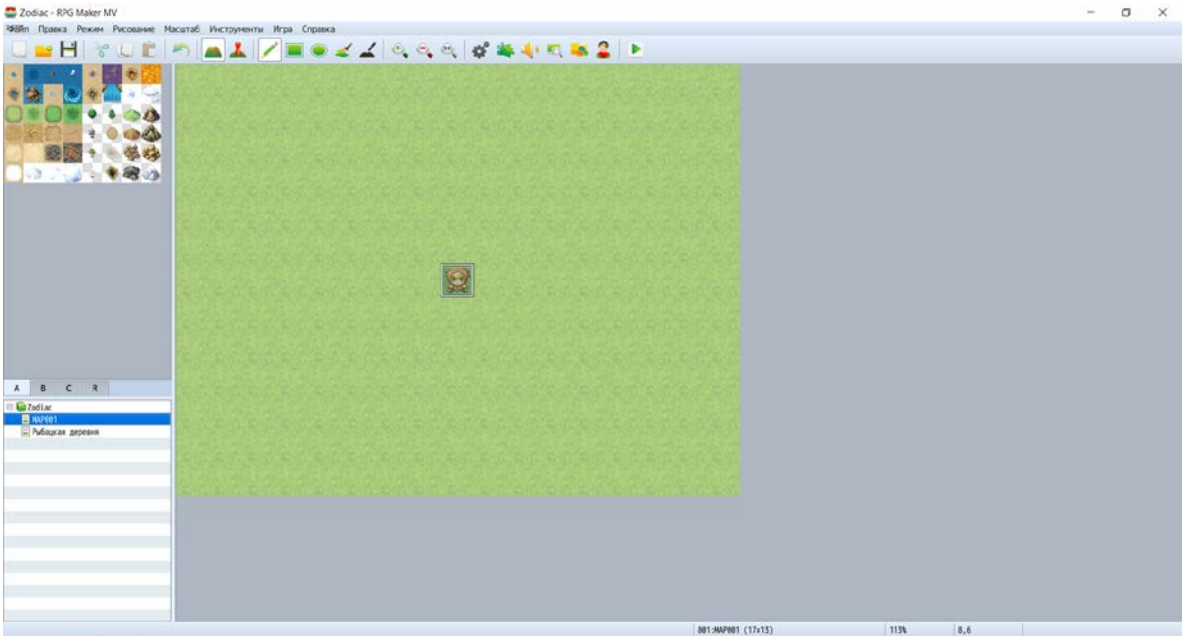

Рисунок 6 — Интерфейс программы RPG Maker MV

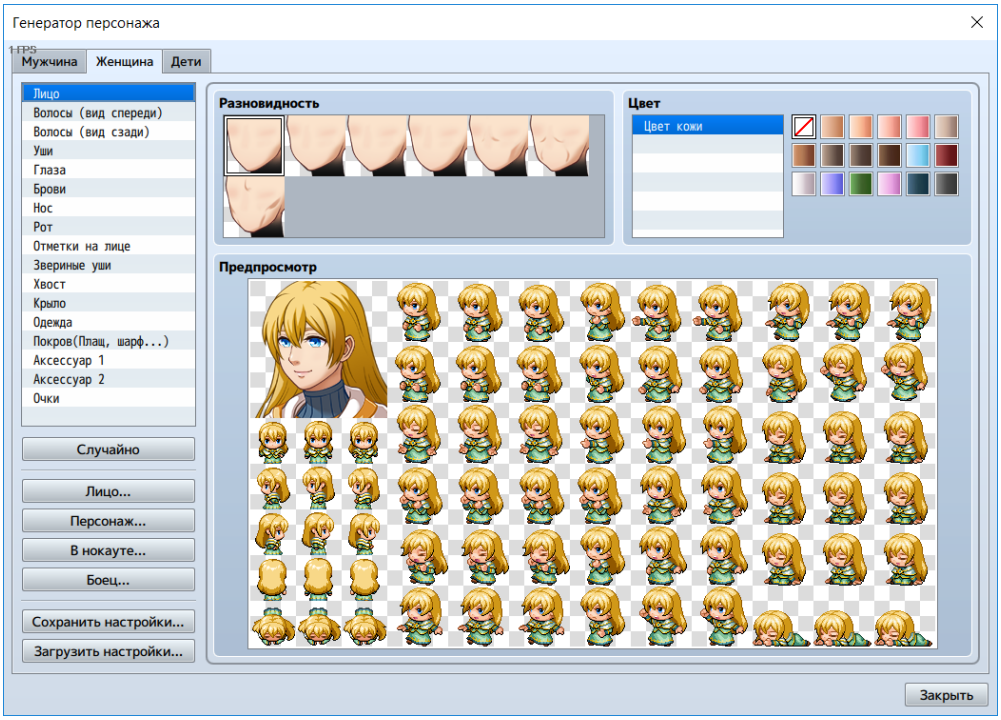

Рисунок 7 — Интерфейс генератора персонажей

1.4 Общий алгоритм реализации ролевых пошаговых игр

Многие люди, играющие в видеоигры, даже не могут представить, сколько трудов и творческих идей вложено в каждую отдельно взятую игру.

Разработка игры — это долгий и сложный процесс, который состоит из различных этапов, включающий в себя технические и творческие моменты. Поэтому, в большинстве случаев, игры создают не отдельные личности, а команды разработчиков. Каждый человек в команде — специалист в своей области знаний.

1.4.1 Цель игры

Первое, что необходимо сделать — определиться с целью разработки. Ответить на вопрос «Что должно получиться в итоге?»

Жанр. Можно с самого начала представлять во всех деталях готовый проект, а можно прямо в процессе разработки додумывать сюжетные составляющие, стиль и некоторые особенности игры. Жанр игры необходимо выбрать в самом начале процесса разработки так как именно он будет основным направлением в развитии игры.

Необходимо проанализировать таблицу жанров и выбрать необходимые для проекта геймплейные элементы. Некоторые элементы могут подарить высокую популярность среди игроков (уничтожение врагов, соревновательные режимы, абсолютно положительный герой, забота о ком-либо), а другие — преданных, придирчивых фанатов (тактика на поле игры, контроль юнитов или городов, уклонение от атак противников), третьи же — отсутствие серьезных конкурентов (обучение, логические задачи, путешествия по различным местам, экономика городов и поселений).

Выбранный жанр можно корректировать по ходу разработки, но его основное ядро должно оставаться прежним.

Сеттинг. Разделение видеоигр на жанры специфично и его сложно сравнивать с системой жанров фильмов или книг. Игровые жанры определяют только основные действия, которые будут совершать игроки в процессе прохождения игры. На вопросы о том где и когда происходят действия игры, отвечает совсем другая характеристика — сеттинг.

Сеттинг — это принадлежность к сюжетной теме или к определённому миру. В среде видеоигр сформировалось несколько самых популярных сеттингов: фэнтези, фантастика, средневековье, стимпанк, пост-апокалипсис, аниме и комиксы.

Выпуск игры в популярном, на данный момент времени, сеттинге обеспечит его популярность, а игроки почувствуют себя комфортно в уже знакомом, по другим проектам, мире. Некоторые видеоигры разрабатываются в своих уникальных сеттингах или в нестандартных сочетаниях популярных тем.

1.4.2 Средства реализации игры

После задания цели игрового проекта, нужно выбрать средства (материалы и инструменты) для её достижения. Программный код как строительный материал — это цифровые изображения, трехмерные модели, звуки и тексты в виде последовательностей единиц и нулей. Код как инструмент — это команды в строчках программного кода, управляющие игровыми объектами всех перечисленных типов.

Программный код. Прежде всего необходимо выбрать язык программирования, который наиболее подходит для реализации проекта. После этого предстоит работа по написанию программного кода, способного оперировать двухмерными или трехмерными объектами в пространстве, привязкой изображений и звуков. Для создания виртуального трехмерного пространства придется использовать сложные геометрические формулы для построения проекции 3D-объектов на плоскость (в уме компьютера объекты существуют

в настоящем трёхмерном пространстве, но для вывода их изображения на плоский двухмерный экран приходится делать пересчеты). По ходу разработки необходимо изучить все форматы изображений и аудиофайлов, всевозможные кодеки и кодировки.

Игровой движок. В наше время можно не заниматься написанием низкоуровневой программной части, а сразу же воспользоваться готовым программным модулем (игровым движком), где уже реализованы базовые функции, способные связать воедино графику, звук, объекты и их движения. Таким образом выбор языка программирования заменяется другой дилеммой — выбором готового игрового движка.

Применение игровых движков ещё не освобождает от использования услуг программистов, но сводит их к минимуму. Стандартный программный модуль придётся настроить под себя, добавить в него что-то своё, чтобы игровой проект получился более уникальным.

1.4.3 Игровая механика

Самая важная творческая часть любой игры — игровая механика. Эта вещь находится не на поверхности, поэтому часто ускользает от взгляда невнимательных ценителей игр.

Молодые подростки (основная часть игровой аудитории) в большинстве своём оценивают игры по качеству графики, и не замечают, что красивые игры хоть и популярны, но их популярность длится всего несколько месяцев после релиза. А в сердцах игроков надолго остаются совсем другие игры, может быть немного неказистые на вид, но имеющие потрясающе захватывающий геймплей. Чем разнообразнее и интереснее игровые возможности, тем дольше игрок остаётся в игре.

Если представить игру в виде живого организма, то игровая механика будет является его нервной системой и головным мозгом. А если представить игру в виде строящегося дома, то игровая механика предстанет как электро-

проводка, трубопровод и прочие бытовые инженерные коммуникации. Игровая механика, по сути своей, это свод правил, по которым будет функционировать игра.

Объекты. Основой всей механики являются игровые объекты. Главный герой игры, компьютерные соперники, второстепенные персонажи (NPC), бонусы, подвижные объекты, декорации — всё это игровые объекты со своими свойствами и возможными действиями.

Управление. Игровая механика определяет какими клавишами будет управляться главный герой или основной игровой объект, какое действие будет происходить после нажатия той или иной кнопки. Сюда же относится законы поведения игровых объектов (физический движок) и поведение врагов (искусственный интеллект).

Физический движок. Если «управление» отвечает за перемещение подконтрольного нам персонажа, то физический движок отвечает за те движения, которые происходят без прямого вмешательства игрока. Эти действия имитируют физические законы реального мира (иногда немного искаженные в сторону фантастики). Брошенный мячик отскакивает от пола, опрокинутая бочка скатывается с наклонной поверхности, выстрел мощным оружием отбрасывает стреляющего назад, хрупкий предмет, брошенный с высоты разбивается — всё это примеры действия физического движка.

В готовых игровых движках чаще всего реализованы и физические движки. Останется присвоить своим уникальным объектам уже готовые физические характеристики: вес, плотность, эластичность, разрушаемость.

Искусственный интеллект (ИИ). ИИ отвечает за поведение компьютерных врагов или союзников.

Роль ИИ значительно разнится в зависимости от жанра игры. В экшенах действия врагов крайне примитивны; в Real Time Strategy (RTS) достаточно пары десятков скриптов, чтобы придать сопернику кажущуюся разумность; в стелс-экшенах, слешерах и файтингах необходимо создать уникальную систему поведения для каждого типа врагов, иначе глупые враги сдела-

ют игру неинтересной. Серьезная стратегическая игра требует колоссальной работы над ИИ, а в простых казуальных играх и в онлайн-проектах, ориентированных на сражения только между реальными игроками, искусственный интеллект вообще не нужен.

1.4.4 Уровни игры

Созданные игровые объекты расставляются в отдельных виртуальных пространствах — уровнях (локациях). Игры чаще всего содержат множество отдельных уровней, переход между которыми происходит по ходу сюжета.

На каждом отдельном уровне расставляются игровые объекты, стенки, платформы, декорации, фоны. Уровни создаются в играх всех жанров. Даже в простой казуальной игре по перестановке цветных камешков есть уровни — в их роли выступают игровые поля и расстановка камней. В браузерных играх в роли локации выступают отдельные html-страницы.

1.4.5 Оформление игры

Далее необходимо улучшать созданную геймплейную часть, с помощью введения более проработанной графики.

Арты. Для начала нужно создать образы героев, врагов, игровых предметов, задних фонов. Первоначально они рисуются либо на бумаге, либо на компьютере с использованием графического планшета. Для небольших игровых студий этот этап не обязателен, но он просто необходим в больших компаниях, чтобы не на пальцах, а на наглядных изображениях объяснить всем дизайнерам, что у них в итоге должно получиться.

2D, 3D модели. На основе артов дизайнеры создают либо двухмерные спрайты из пикселей, либо трёхмерные модели из полигонов.

Анимации. Для игровых объектов, которые будут передвигаться в ходе игры, создаются анимации.

В настоящее время, для создания человекоподобных 3D-персонажей существует специальная технология «Motion Capture», позволяющая создавать анимации на основе движений настоящих людей. Эта технология доступна лишь крупным и очень богатым компаниям. Для использования Motion Capture нужно не только приобрести дорогостоящее оборудование, но ещё и нанять группу актёров, с которых будут записываться движения.

Фоны. С задними фонами всё гораздо проще — нарисовать один раз и поставить в нужное место на уровне без всяких изменений.

Спецэффекты. Визуальные спецэффекты — это те же анимации, только вместо перемещения объектов в них используются перемещения частиц и светофильтров. Лучи света в разные стороны при взятии бонусов, огонь на горящем здании, дымовая завеса после взрыва гранаты, лазерные лучи из дула винтовок, наложение фильтров размытия при нахождении под водой и фильтров затемнения в плохо освещённых местах — всё это спецэффекты. Без подобных эффектов игра будет казаться пресной и слишком обыденной. Использование спецэффектов добавляет игре яркости, сочности и экспрессивности.

Оформление экрана и меню. Оформление необходимо не только для игровых уровней, но и для системы, которая объединяет их в единое целое — игровое меню. Начальное меню — это визитная карточка игры, и она должна выглядеть приятно для взора игрока. На самом экране игры так же есть много элементов, к которым применяется оформление — количество жизней, инвентарь героя, списки заданий, экраны диалогов. На английском языке всё это подходит под сокращение — GUI (Graphic User Interface — графический пользовательский интерфейс).

1.4.6 Сюжет игры

Привлечь игрока к игре довольно сложно, но ещё сложнее сделать так, чтобы игрок остался в игре до конца. Любое разочарование или трудное ме-

сто может быстро оттолкнуть игрока от дальнейшего прохождения игры. В большинстве случаев игрок оставит игру без сожаления. Только грамотно поданный сюжет может заставить игрока пройти игру, а значит — дослушать историю до конца.

Когда компьютерные игры только начинали создаваться, они обходились без сюжета, затягивая к себе игроков лишь своим игровым процессом. Но на сегодняшний день даже к самой простой игре добавляют сюжет, не говоря уже о крупных игровых проектах AAA-класса (неформальный термин, обозначающий класс высокобюджетных компьютерных игр).

Существование в игре сюжета лишь для того, чтобы он был, не даёт положительного эффекта, но разработчики продолжают придумывать новые истории, не отличающиеся особой оригинальностью. Сюжет приносит пользу лишь в том случае, если он может задеть чувства игрока. Для этого нужна уникальность, качественность и правдоподобность сюжета; каждый герой должен иметь свою личность, и совершать поступки согласно ей; действующих лиц и событий должно быть не больше, чем может воспринять человеческий мозг, иначе сюжет превратится в сложный для восприятия объект; события должны происходить логично (загадочность и туманность нужны для поддержания сюжетной интриги, но при их создании должна быть логичность).

Скрипты, события. Самый лучший вариант — когда сюжет существует прямо внутри игры. Это достигается с помощью использования скриптовых сценок.

Скрипт представляет собой следующее: игрок заходит в определённое место, или совершает нужное действие, или выполняются ещё какие-то необходимые условия, и после этого начинают выполняться действия, запрограммированные вами на этот случай. С помощью скриптовых событий можно вносить разнообразие в игровой процесс или даже превратить игру в аттракцион нескончаемых скриптов. Единственный минус такого способа —

у игрока уменьшается свобода действий. Всё происходит по воле скриптов, и мало зависит от действия игрока.

Диалоги, повествования. В старых классических играх сюжет существует обособленно от игрового процесса. Например, при загрузке или окончании уровней знакомят с сюжетной историей, рассказывают об отношениях между героями и врагами, объясняют, что и для чего нужно сделать на уровне. В процессе самой игры ничего из вышесказанного не имеет никакого значения, и игрок может смело пропускать все эти тексты. Чаще всего так и происходит — тексты остаются не прочитанными. А всё из-за того, что нет никакой веской причины их читать.

Другое дело — внутриигровые тексты или диалоги. Они происходят в процессе игры, но в безопасных местах, или с остановкой игрового времени, чтобы игрок мог сосредоточиться только на тексте. Повествования игроку приходится выслушивать, так как игра на это время замирает, но не останавливается совсем. А в диалогах ещё и нужно выбирать вариант ответа. Выбор варианта придаёт прослушиванию текста интерактивность и хоть какой-то практический смысл — правильно выбранный ответ может принести дополнительный бонус, облегчить дальнейшую игру или сохранить выбранный характер героя в ролевых играх.

При создании игры тексты повествований и диалогов лучше хранить в отдельных файлах, подгружаемых по ходу игры. Отделение художественного текста от технических кодов поможет в будущем, если будет принято решение создать локализованную версию игры на других языках мира.

1.4.7 Звуковое оформление игры

Звук в игре — одна из основных составляющих игры, добавляющая реалистичности и необходимая для полноты игрового опыта. Очень важно, чтобы звук в игре присутствовал.

Звуковые эффекты. Для любого игрового движения нужно добавить соответствующий звук. Это могут быть удары меча, нанесение рукопашного удара, звуки движения автомобиля, получение бонуса.

Хорошие звуковые эффекты не только заполняют тишину, но и являют собой продолжение графического стиля игры. Веселая аркада наполнена не менее весёлыми мелодичными звуками, спортивный автосимулятор наполнен рокочущим гулом моторов и лязгом тормозов, трехмерный боевик оглушает пулемётными очередями, падающими гильзами и клацающими затворами винтовок.

Чаще всего в качестве звуковых эффектов используются реальные звуки, записанные в цифровом виде.

Музыка. Кроме звуков для полноценной игры нужна и музыка (саундтрек). Она будет звуковым фоном для происходящего на экране. Музыка так же является одним из стилистических элементов игры, и сильнее всего влияет на настроение игрока. Готовую музыку нужно долго выбирать по подходящему темпу и настроению [10].

1.4.8 Тестирование и корректировка игры

Процесс разработки большой игры построен таким образом, что различными её элементами занимаются различные специалисты. На начальном этапе игра представляет собой разрозненный набор творческих наработок в различных областях искусства: изображения, звуки, 3D-модели, архитектура, тексты, сценки, видеовставки, оформление. С помощью программных средств разрозненные объекты соединяются в единую сложную систему.

Сведение материала (α-версия). При построении игры на игровом движке объединение объектов происходит постепенно с самого начала процесса. Пока игра не собрана до конца, её называют альфа версией. В этот момент уже можно заниматься тестированием отдельных уровней, скриптов и прочих механизмов.

На этом этапе уже технически возможно выпустить демоверсию или видеоролик с игровым процессом, чтобы заранее привлечь игроков к игре.

Устранение ошибок (β-версия). Когда игра полностью собрана, остаётся лишь устранить получившиеся ошибки (bugs). Они появляются в любом случае, так как игра — это система со сложной структурой.

Поиском ошибок в игре занимаются тестеры. Очень часто в качестве тестеров привлекаются группы обычных игроков, и это служит началом их карьеры в игровой индустрии. Проще всего эта проблема решается в онлайн-играх — разработчики организовывают открытые бета-тесты, в которых участвуют вообще все желающие игроки.

Цели, с которыми производится тестирование программного обеспечения, вполне очевидны. Разумеется, первостепенная задача тестеров — выполнить наглядную демонстрацию того, что продукт действительно работает. Это необходимо как самим разработчикам, так и издателю. От мнения издателя зависит, попадет программа к конечному потребителю или нет. Еще одна не менее важная задача тестирования — определение возможных изъянов разработанного продукта. Речь идет, в первую очередь, о ситуациях, в которых программное обеспечение может повести себя ненадлежащим образом.

В ходе тестирования программного обеспечения, как и при любых других способах проверки продукции, определяется его уровень качеств. Тот, кто тестирует приложение, должен убедиться в том, что объект соответствует всем предъявляемым к нему требованиям.

2 РАЗРАБОТКА ТЕХНИЧЕСКОЙ ДЕМОНСТРАЦИОННОЙ ВЕРСИИ МУЛЬТИМЕДИЙНОГО ПРОДУКТА В ЖАНРЕ «ЯПОНСКАЯ РОЛЕВАЯ ПОШАГОВАЯ ИГРА»

2.1 Потенциальная аудитория потребителей мультимедийного продукта

Проект ориентирован на аудиторию поклонников жанра японских ролевых игр. Он будет интересен как детям, так и взрослым, которые увлекаются играми подобного жанра. Одно из главных достоинств проекта — это взаимосвязь боевой системы японских ролевых игр и системы квестов европейского образца [12].

Проанализировав целевую аудиторию игры, было принято решение, что продукт должен обязательно иметь сюжет, предлагать игроку разные механики (сдача квестов, «Пошаговая» боевая система, взаимодействие с инвентарём), локации должны быть устроены так, чтобы по ним можно было ходить в любом направлении, а не идти из одного пункта в другой по строгому маршруту. Для поддержания атмосферы нужно использовать подходящее звуковое сопровождение и правильно выставленный свет.

2.2 Постановка задачи мультимедийного продукта

2.2.1 Актуальность мультимедийного продукта

JRPG обладают большим количеством поклонников, и подобные игры хорошо зарекомендовали себя в качестве коммерческого продукта и доступны на таких платформах, как Steam. Благодаря популярности, выходящих до

сих пор, игр в данном жанре данный проект имеет высокую актуальность для создания.

2.2.2 Цель и назначение мультимедийного продукта

Продукт создается с целью познакомить новичка с основными механиками жанра, а ветеранам подарить новые ощущения от уникальных гемплейных составляющих.

Игра предназначена для заполнения свободного времени, но в отличие от просмотра фильма или прочтения книги, даёт возможность человеку почувствовать себя причастным к событиям происходящим в сюжете, а также испытать различные эмоции и отвлечься от рутины.

Игра должна вызывать эмоциональный отклик и запомниться игроку, чтобы привлечь его внимание к разработчику и его последующим играм.

2.2.3 Требования к мультимедийному продукту

Для проекта были сформулированы следующие требования:

1. Локации должны быть выполнены в средневековом стиле.
2. Боевая система и инвентарь должны соответствовать жанру.
3. Должны быть реализованы анимации персонажа и врагов.
4. Система квестов должна соответствовать европейскому стандарту.
5. Звуки и музыка должны соответствовать миру игры.
6. Должно быть реализовано главное меню.
7. После включения игры должно быть видео с лого разработчика.
8. Должна быть реализована возможность возврата в главное меню.
9. Должны быть реализованы границы карты, чтобы у игрока не было возможности уйти за её пределы.
10. Должна присутствовать настройка громкости музыки и звука.

2.2.4 Входные данные к мультимедийному продукту

Для фоновой музыки использовались аудиофайлы в формате *.ogg — три основные темы для локаций, тема для сцены боя, звуки победы, поражения и ударов.

Среди входных данных к проекту были графические элементы навигации, текстуры земли, травы, 3D-модели домов, деревьев, и камней. Некоторые модели были взяты из Интернета, некоторые — из магазина ресурсов Unity.

2.2.5 Характеристики оборудования для реализации мультимедийного продукта

Чтобы разработать мультимедийный продукт в жанре японской ролевой пошаговой игры, необходим компьютер с характеристиками, которые намного превышают характеристики компьютера среднестатистического пользователя. Это необходимо, так как программы для разработки подобных мультимедийных продуктов очень требовательны.

Для реализации проекта был использован ноутбук DELL G5 с характеристиками, указанными в таблице 3.

Таблица 3 — Характеристики оборудования

Характеристика	Описание
Процессор	Intel(R) Core(TM) i7-8750H CPU: 2,2 ГГц
Количество ядер процессора	6
Операционная система	Windows 10
Тип системы	64-разрядная операционная система
Размер оперативной памяти	16,00 Гб
Частота оперативной памяти	2400 МГц
Модель дискретной видеокарты	NVIDIA GeForce 1060
Объём видеопамяти	2,00 Гб
Дисплей	15,6” с разрешением Full HD (1920 x 1080)

2.3 Жизненный цикл мультимедийного продукта

2.3.1 Этап эскизного проектирования

В качестве примера для интерфейса инвентаря была выбрана игра Chrono trigger. На её основе был создан эскиз будущего инвентаря (рисунок 8). Данная система является ярким примером того, как должен быть устроен инвентарь в игре подобного жанра. В основном интерфейсе находится окно с описанием доступных персонажей и их характеристиками. Также, в этом окне осуществляется выбор следующего действия. На втором окне осуществляется выбор предмета из тех, которые находятся в инвентаре у игрока. На третьем окне осуществляется выбор героя, на которого будет применяться предмет.

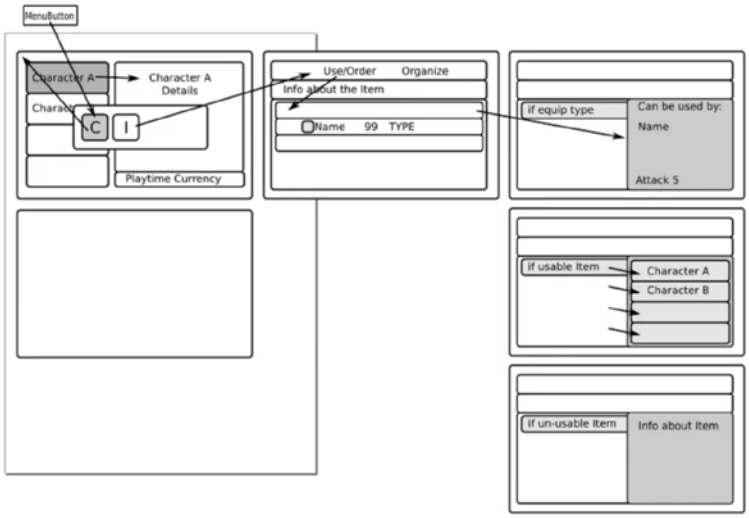

Рисунок 8 — Эскиз интерфейса инвентаря

Для боевой системы, как референс, была взята система из игры Final Fantasy 9. Бой осуществляется следующим образом — у каждого участника сражения есть своя шкала действия. Персонаж осуществляет атаку, когда его шкала действия полностью заполняется. Враги не ждут, когда игрок сделает свое действие и нападают в каждом удобном случае. Среди атак есть различие на магические и физические атаки, что делает бой разнообразнее (рисунок 9).

Рисунок 9 — Боевая система Final fantasy 9

Как пример для реализации квестовой системы был принят стандарт в европейских ролевых играх. За основу, в таких системах, взят принцип «получи и принеси». Такой принцип является хорошим примером для реализации, так как подобного смешения геймплейных составляющих замечено и очень малого количества игр жанра, в котором разрабатывается продукт (рисунок 10).

Рисунок 10 — Квестовая система в игре «Скайрим»

Эскиз первой локации был разработан в программе RPG Maker MV (рисунок 11).

Рисунок 11 — Эскиз локации «Рыбацкая деревня»

2.3.2 Этап разработки элементов дизайна мультимедийного продукта

Отталкиваясь от основного эскиза инвентаря, который был взят за основу, был разработан интерфейс, представленный на рисунках 12 и 13.

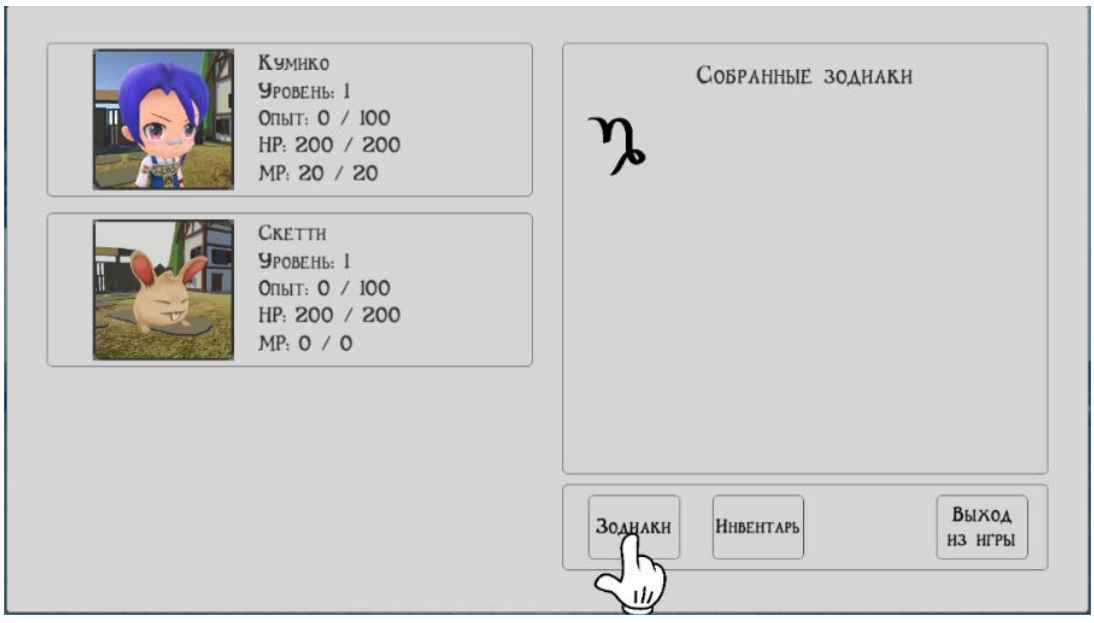

Рисунок 12 — Начальная панель инвентаря

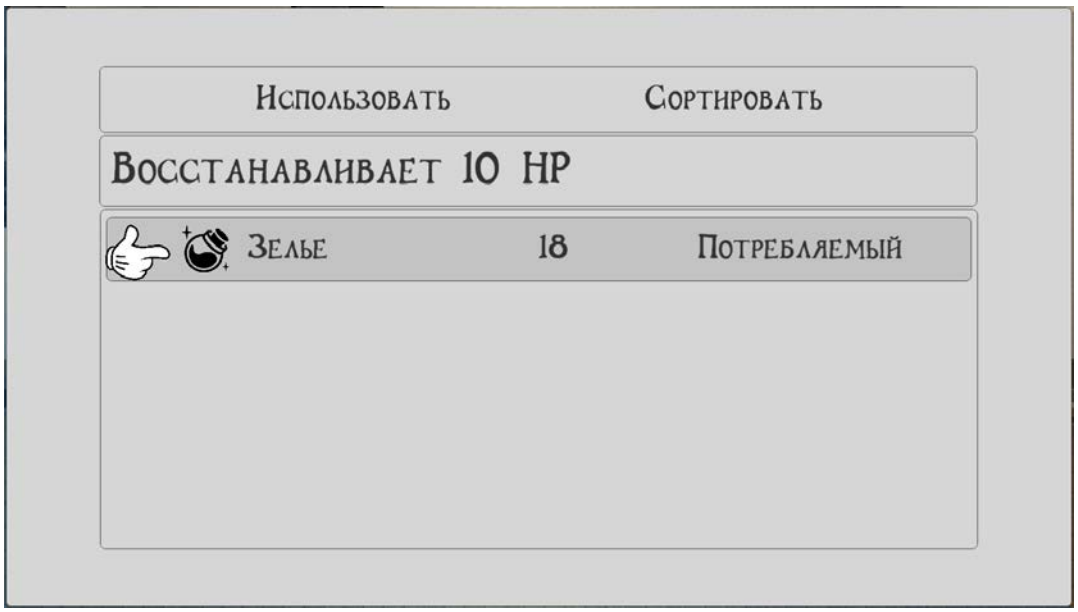

Рисунок 13 — Страница предметов

Эскиз локации был переработан в программе 3Ds Max, для создания карты высот, которая являлась первоначальной картой из примитивов (рисунок 14).

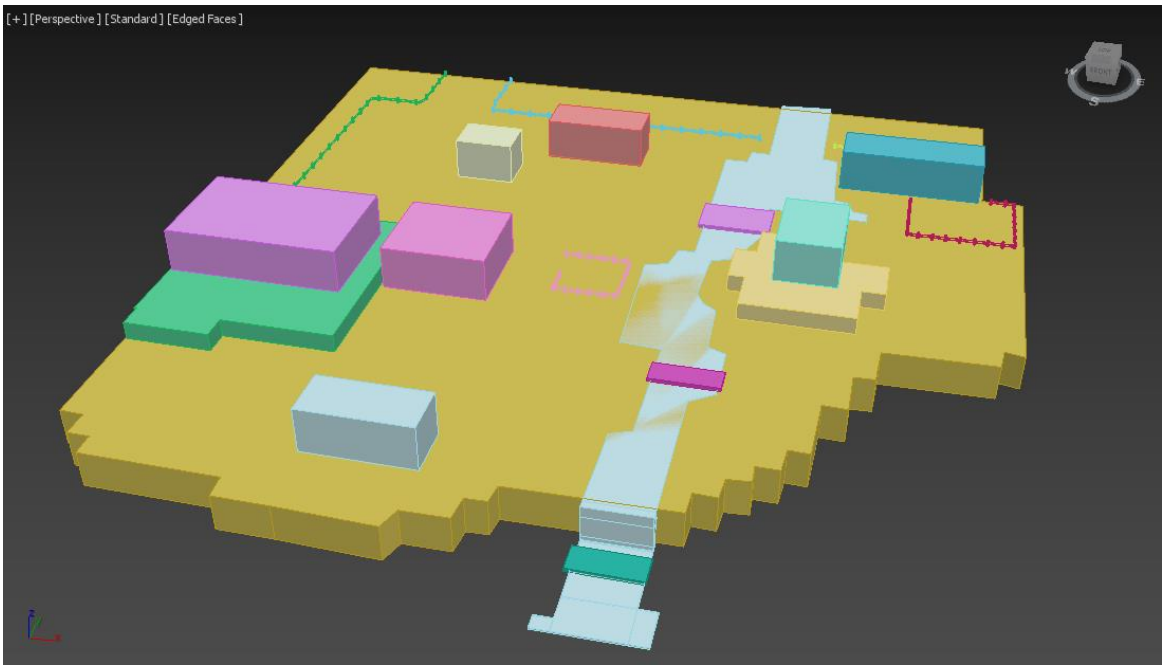

Рисунок 14 — Карта высот локации «Рыбацкая деревня»

Отталкиваясь от выбранного примера, была разработана боевая система (рисунок 15).

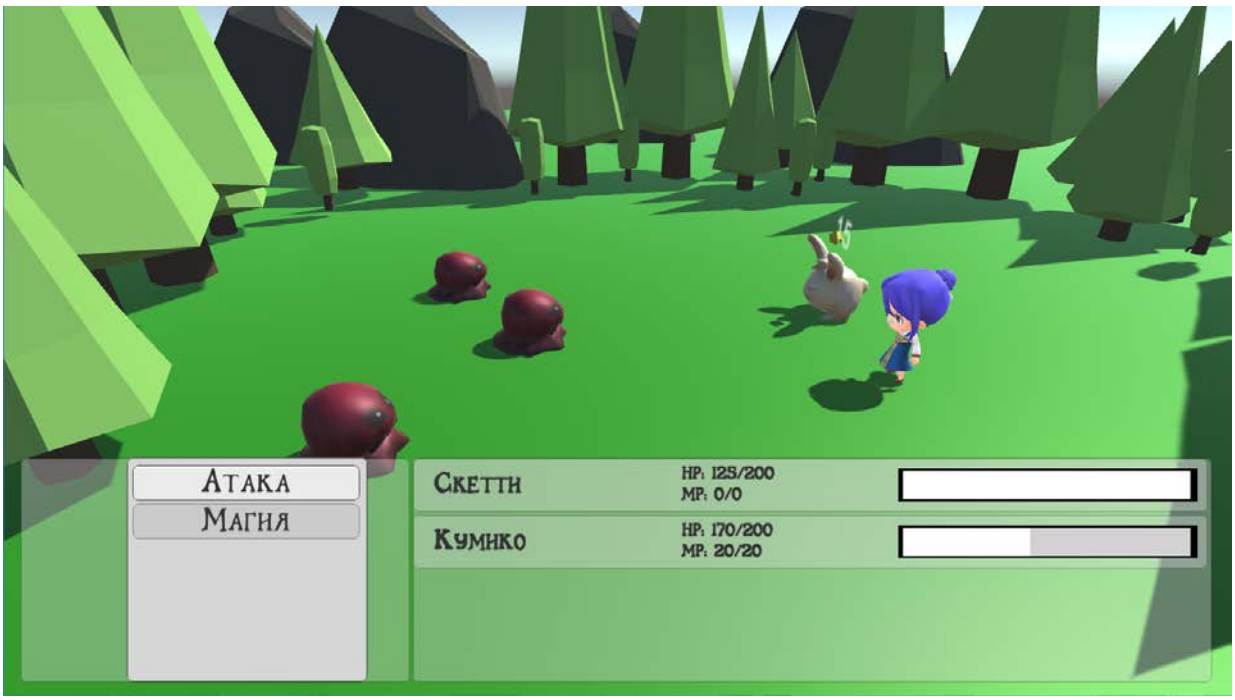

Рисунок 15 — Боевая система в создаваемом проекте

Придерживаясь стандарта, выявленного из рассматриваемого примера, был разработан интерфейс квестовой системы (рисунок 16).

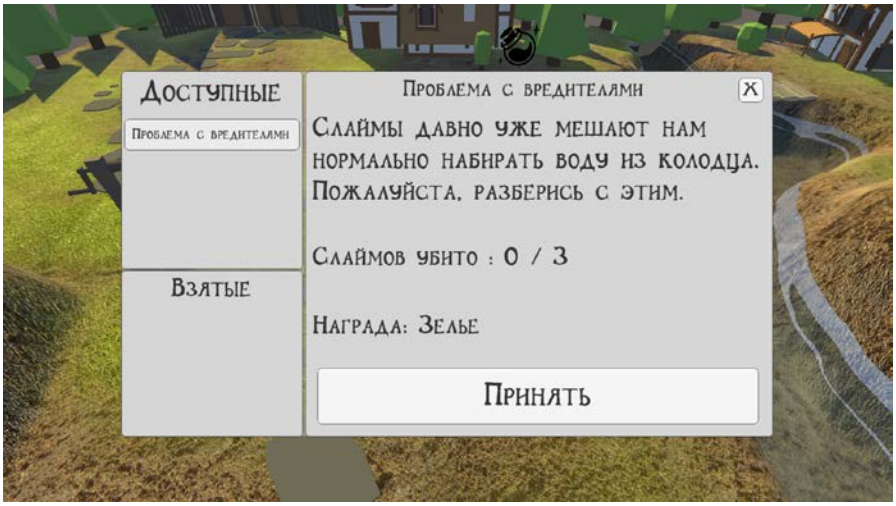

Рисунок 16 — Графический интерфейс пользователя для квестовой системы

2.3.3 Программирование боевой системы

Программирование в Unity Engine происходит посредством создания скриптов и присвоения их к объектам. В начале боя на сцену добавляются монстры и игроки, а затем принимается решение о том, в каком порядке они совершают ходы.

Боевой поток управляется с использованием конечного автомата с двумя основными состояниями; первое состояние, чтобы пометить действия, второе состояние, чтобы выполнить действие, когда придет время. Также, существуют ещё четыре состояния, два из которых отвечают за само воспроизведение действия и на проверку — жив ли персонаж или враг (рисунок 17).

```
public class BattleStateMachine : MonoBehaviour
{
    22 references
    public enum PerformAction
    {
        WAIT,
        TAKEACTION,
        PERFORMACTION,
        CHECKALIVE,
        WIN,
        LOSE
    }
```

Рисунок 17 — Состояния автомата боя

Также, основной автомат боя отвечает за создание элементов интерфейса и их привязку к определенным функциям, по которым персонажи будут понимать, какое именно действие они должны совершить (рисунок 18).

```
1 reference
void CreateAttackButtons()
{
    GameObject AttackButton = Instantiate(actionButton) as GameObject;
    Text AttackButtonText = AttackButton.transform.Find("Text").gameObject.GetComponent<Text>();
    AttackButtonText.text = "Атака";
    AttackButton.GetComponent<Button>().onClick.AddListener(()=>Input1());
    AttackButton.transform.SetParent(actionSpacer, false);
    atkBtns.Add(AttackButton);

    GameObject MagicAttackButton = Instantiate(actionButton) as GameObject;
    Text MagicAttackButtonText = MagicAttackButton.transform.Find("Text").gameObject.GetComponent<Text>();
    MagicAttackButtonText.text = "Магия";
    MagicAttackButton.GetComponent<Button>().onClick.AddListener(() => Input3());
    MagicAttackButton.transform.SetParent(actionSpacer, false);
    atkBtns.Add(MagicAttackButton);

    if(HeroesToManage[0].GetComponent<HeroStateMachine>().chara.hero.MagicAttacks.Count > 0)
    {
        foreach (BaseAttack magicAtk in HeroesToManage[0].GetComponent<HeroStateMachine>().chara.hero.MagicAttacks)
        {
            GameObject MagicButton = Instantiate(magicButton) as GameObject;
            Text MagicButtonText = MagicButton.transform.Find("Text").gameObject.GetComponent<Text>();
            MagicButtonText.text = magicAtk.attackName;
            AttackButton ATB = MagicButton.GetComponent<AttackButton>();
            ATB.magicAttackToPerform = magicAtk;
            MagicButton.transform.SetParent(magicSpacer, false);
            atkBtns.Add(MagicButton);
            int curMP = (int)HeroesToManage[0].GetComponent<HeroStateMachine>().chara.hero.curMP;

            if (magicAtk.attackCost <= curMP)
            {
                MagicButton.GetComponent<Button>().interactable = true;
            }
        }
    }
    else
    {
        MagicAttackButton.GetComponent<Button>().interactable = false;
    }
}
```

Рисунок 18 — Функция, отвечающая за создание основных кнопок атаки

Состояние принятия решения для игрока приведет к появлению меню с устойчивыми для JRPG параметрами — Атака и Магия. Как только игрок принимает решение о действии, действие принятия решения удаляется из очереди и запускается счетчик ожидания до принятия нового (рисунок 19).

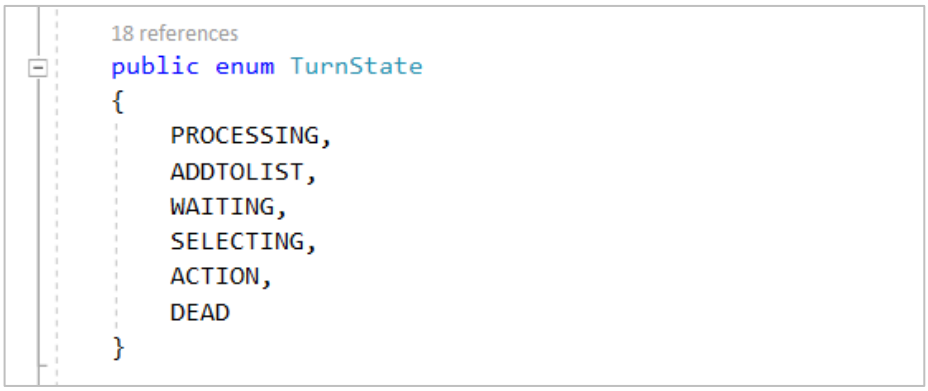

Рисунок 19 — Состояния автомата игрока

Состояние принятия решения для врага проверяет сцену и решает, что делать дальше, а затем оно также удалит его действие принятия решения и запустит счетчик ожидания до принятия нового решения (рисунок 20).

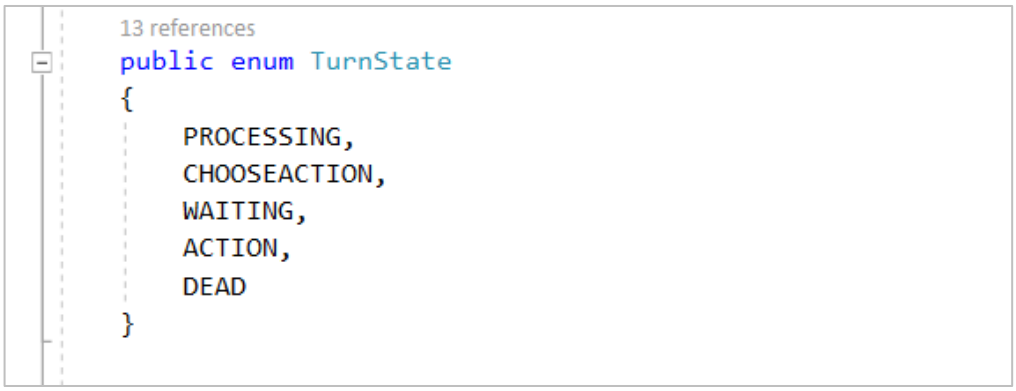

Рисунок 20 — Состояния автомата врага

Также, у автоматов игрока и врага есть состояния, взаимодействующие с автоматом боевой системы. Это такие состояния, как ожидание, совершение действия и смерть.

2.3.4 Программирование квестов

Система квестов реализована по системе получения определенного предмета, чтобы завершить квест. Данным предметом может быть все, что

угодно, заданное специальной переменной. Сам квест имеет автомат состояний, который сообщает пользовательскому интерфейсу о доступности квеста (рисунок 21).

```
public enum QuestProgress
{
    NOT_AVALIABLE,
    AVALIABLE,
    ACCEPTED,
    COMPLETE,
    DONE
}
```

Рисунок 21 — Состояния автомата квеста

У пользовательского интерфейса есть свой скрипт-менеджер, отвечающий за его правильное отображение при контакте с квестовым персонажем или при просмотре активных квестов на специальной панели (рисунок 22).

```
public void ShowSelectedQuest(int questID)
{
    for(int i = 0; i < avaliableQuests.Count; i++)
    {
        if(avaliableQuests[i].id == questID)
        {
            questTitle.text = avaliableQuests[i].title;
            if(avaliableQuests[i].progress == Quest.QuestProgress.AVALIABLE)
            {
                questDescription.text = avaliableQuests[i].description;
                questReward.text = "Награда: " + avaliableQuests[i].itemReward.itemName;
                questSummary.text = avaliableQuests[i].questOblective + " : " + avaliableQuests[i].questObjectiveCount + " / " + avaliableQuests[i].questObjectiveRequirement;
            }
        }
    }

    for(int i = 0; i < activeQuests.Count; i++)
    {
        if(activeQuests[i].id == questID)
        {
            questTitle.text = activeQuests[i].title;
            if (activeQuests[i].progress == Quest.QuestProgress.ACCEPTED)
            {
                questDescription.text = activeQuests[i].hint;
                questReward.text = "Награда: " + activeQuests[i].itemReward.itemName;
                questSummary.text = activeQuests[i].questOblective + " : " + activeQuests[i].questObjectiveCount + " / " + activeQuests[i].questObjectiveRequirement;
            }
            else if (activeQuests[i].progress == Quest.QuestProgress.COMPLETE)
            {
                questDescription.text = activeQuests[i].congratulation;
                questReward.text = "Награда: " + activeQuests[i].itemReward.itemName;
                questSummary.text = activeQuests[i].questOblective + " : " + activeQuests[i].questObjectiveCount + " / " + activeQuests[i].questObjectiveRequirement;
            }
        }
    }
}
```

Рисунок 22 — Функция, отвечающая за вывод информации о квесте

У самого квеста, также есть набор переменных, которые содержат информацию о том, что необходимо сделать для его выполнения. На рисунке 23 показаны все переменные с описанием функций, которые они выполняют для стабильной работы квеста.

```
public string title;                    //Название
public int id;                          //ID квеста
public QuestProgress progress;          //Состояние квеста
public string description;              //Описание квеста
public string hint;                     //Подсказка
public string congratulation;           //Сообщение о выполнении
public string summery;                  //Итог
public int nextQuest;                   //ID следующего в цепи квеста

public GameObject itemToActive;         //Предмет, который необходимо активировать

public string questOblective;           //Цель (предмет) квеста
public int questObjectiveCount;         //Количество имеющихся предметов для выполнения квеста
public int questObjectiveRequirement;   //Нужное количество предметов для выполнения квеста

public int expReward;                   //Награда опыт
public Item itemReward;                 //Награда предмет
```

Рисунок 23 — Переменные квеста

Среди переменных квеста есть одна, которая отвечает за предмет для активации. Данный предмет будет активироваться только при взятии определенного квеста. Заполняются все переменные внутри движка, после присвоения менеджера к объекту (рисунок 24).

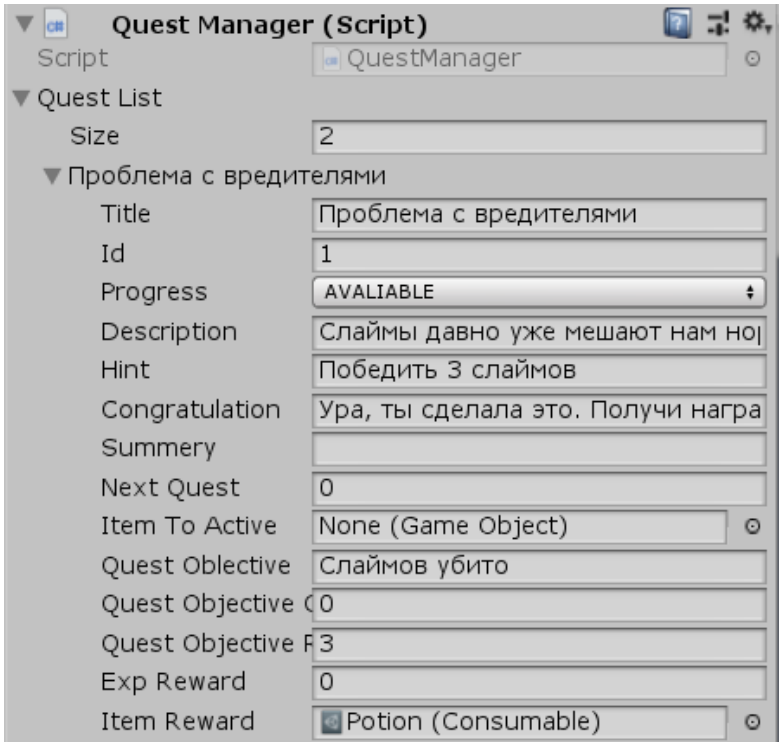

Рисунок 24 — Менеджер квестов на объекте в Unity Engine

Это необходимо для появления в игровом мире различных предметов, которые необходимы для выполнения квеста, либо должны появиться в связи с сюжетом.

2.3.5 Программирование инвентаря

У инвентаря также реализован автомат, содержащий различные состояния (рисунок 25). Каждое из этих состояний отвечает за активацию или деактивацию определенной панели, содержащей функции для отображения информации или предметов.

```
public enum InventoryStates
{
    DISABLED,
    OPTIONS_PANEL,
    ZODIAC_PANEL,
    INVENTORY_PANEL,
    INVENTORY_USE,
    USE_PANEL
}
```

Рисунок 25 — Состояния автомата инвентаря

В панели инвентаря отображаются предметы определенного класса, которые разделяются на несколько видов — расходуемые, экипировка, зодиаки. Управление в инвентаре реализовано в соответствии с рассматриваемым примером — через кнопки на клавиатуре (рисунок 26). Действия привязаны к стандартным кнопкам управления, которые используются в большинстве компьютерных видеоигр.

```
if(Input.GetKeyDown(KeyCode.A) && currentOption > 0)
{
    source.Play();
    currentOption--;
}
else if(Input.GetKeyDown(KeyCode.D) && currentOption < optionsList.Count)
{
    source.Play();
    currentOption++;
}

if((Input.GetKeyDown(KeyCode.Space) || Input.GetKeyDown(KeyCode.Mouse0)) && currentOption == 0)
{
    source.Play();
    InventoryUI.instance.zodDescriptionTextField.text = Inventory.instance.zod.items[zodCurrentOption].item.description;
    inventoryState = InventoryStates.ZODIAC_PANEL;
}
else if((Input.GetKeyDown(KeyCode.Space) || Input.GetKeyDown(KeyCode.Mouse0)) && currentOption == 1)
{
    source.Play();
    inventoryState = InventoryStates.INVENTORY_PANEL;
}
else if((Input.GetKeyDown(KeyCode.Space) || Input.GetKeyDown(KeyCode.Mouse0)) && currentOption == 2)
{
    source.Play();
    Pause.instance.Exit();
}

if (Input.GetKeyDown(KeyCode.Backspace) || Input.GetKeyDown(KeyCode.Mouse1))
{
    source.Play();
    inventoryState = InventoryStates.DISABLED;
}
```

Рисунок 26 — Код кнопок управления в инвентаре

Посредством выбора определенного элемента, который отвечает за тот или иной предмет, происходит использование и применение эффекта данного предмета на выбранного персонажа. У класса предмета реализован перезаписываемая функция, которая отвечает за использование, чтобы каждый из типов предметов использовался по-своему (рисунок 27).

```
public override void UseItem(Hero chara, GameObject errorPanel)
{
    if (chara.hero.curHP<chara.hero.baseHP && isPotion)
    {
        chara.hero.curHP += hpAmount;
        if(chara.hero.curHP > chara.hero.baseHP)
        {
            chara.hero.curHP = chara.hero.baseHP;
        }
    }
    else if(chara.hero.curHP == chara.hero.baseHP && isPotion)
    {
        errorPanel.SetActive(true);
    }

    if(chara.hero.curMP<chara.hero.baseMP && isEther)
    {
        chara.hero.curMP += mpAmount;
        if (chara.hero.curMP > chara.hero.baseMP)
        {
            chara.hero.curMP = chara.hero.baseMP;
        }
    }
    else if (chara.hero.curHP == chara.hero.baseHP && isEther)
    {
        errorPanel.SetActive(true);
    }
}
```

Рисунок 27 — Функция использования расходного предмета

При получении предмета, информация о нем заносится в базу предметов, а при его использовании, данные о нем удаляются, либо, если предметов больше одного, уменьшается на единицу (рисунок 28).

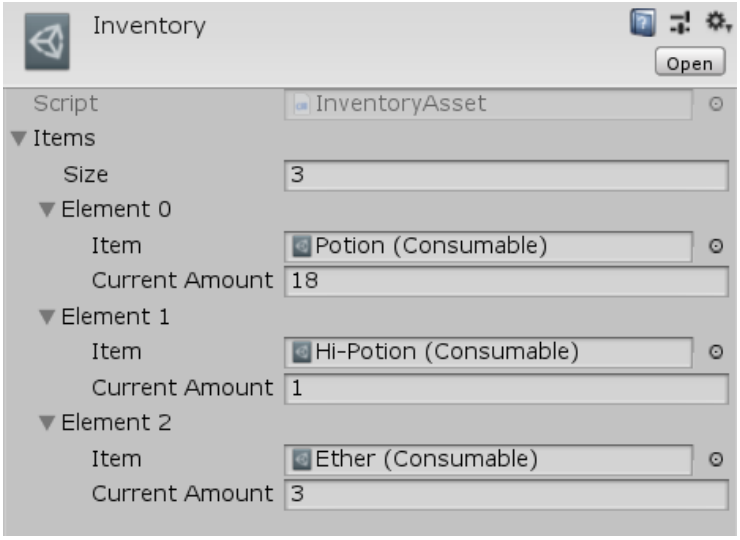

Рисунок 28 — База предметов внутри движка

2.3.6 Разработка моделей персонажей и врагов

За основу главного и квестового персонажей была взята бесплатно распространяемая модель «Haruko» из официального магазина Unity Asset Store (рисунок 29).

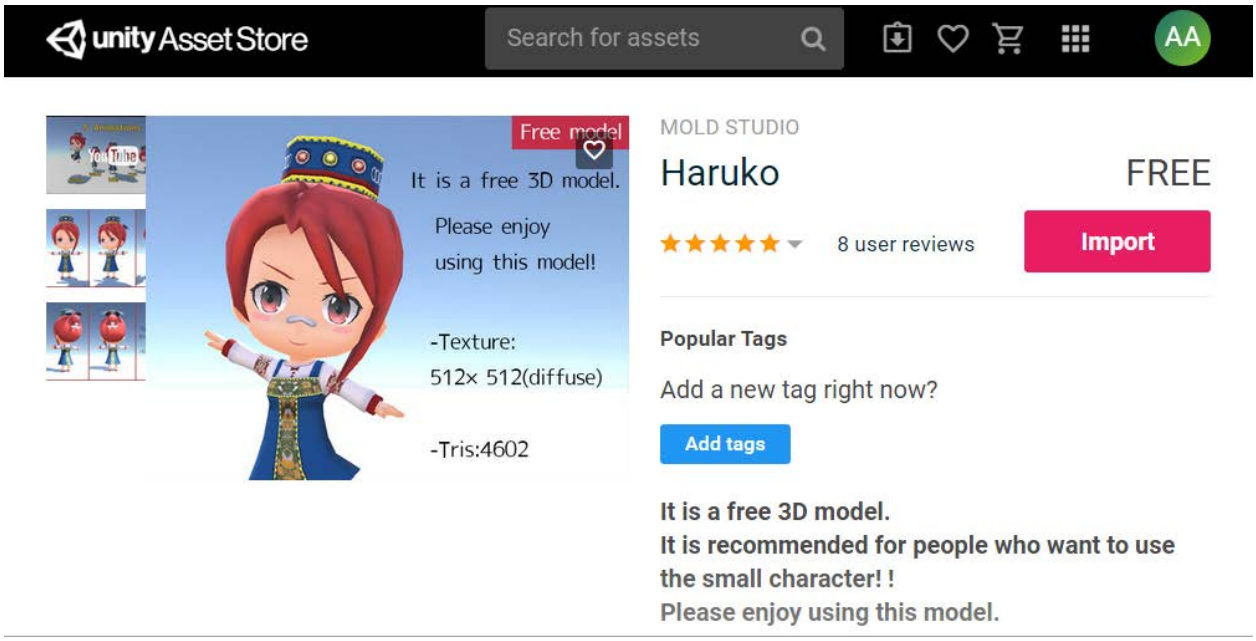

Рисунок 29 — Страница модели в Unity Asset Store

Так как внешний вид данной модели не полностью соответствовал необходимым для главного героя критериям, модель подверглась изменениям в программах 3Ds Max и Adobe Photoshop.

Была удалена шапочка с головы модели (рисунок 30).

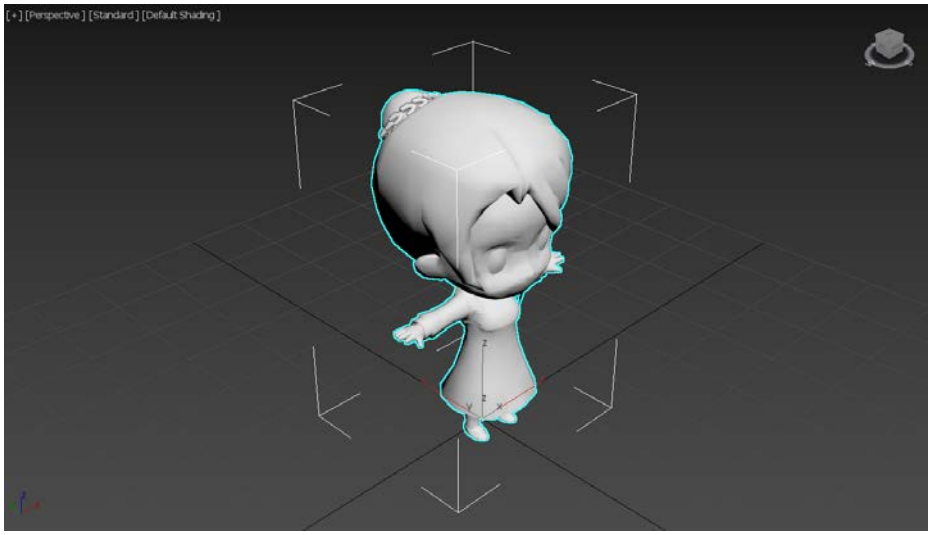

Рисунок 30 — Измененная модель персонажа в 3Ds Max

Был изменен цвет волос на развертке модели, с помощью функции замены цвета (рисунок 31).

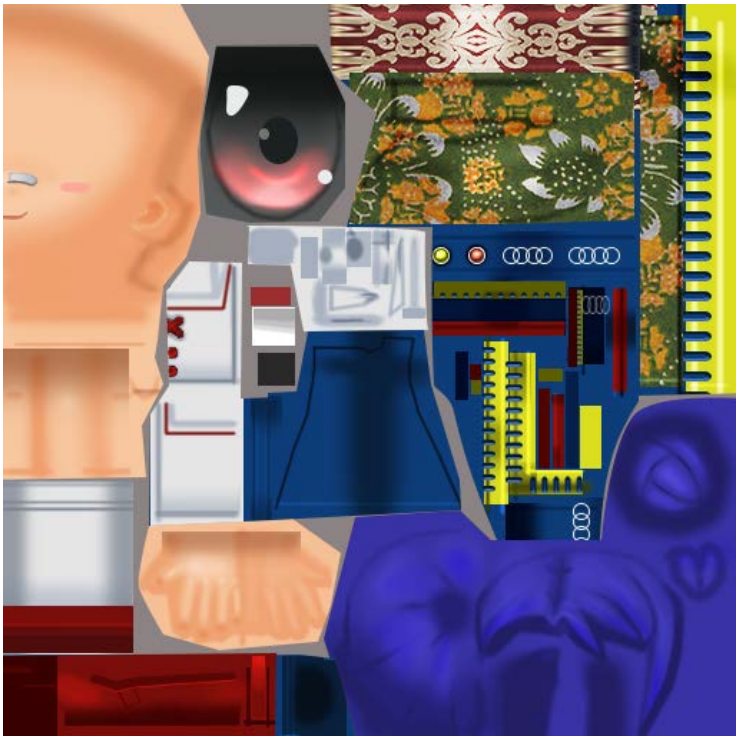

Рисунок 31 — Развертка персонажа с измененным цветом волос

Квестовым персонажем является сама модель «Haruko», не подвергнутая никаким изменениям. Анимации для модели предоставлялись вместе с ней и их достаточно было только настроить в движке (рисунок 32).

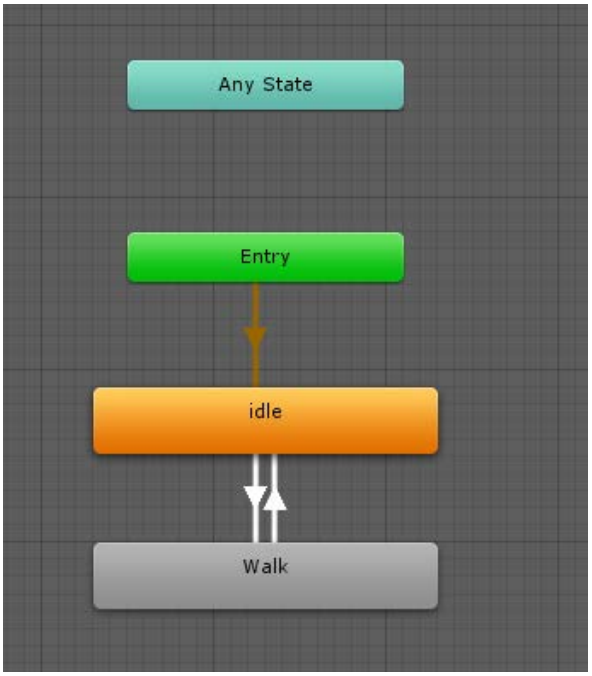

Рисунок 32 — Дерево анимации для состояния покоя и передвижения

Но, анимаций для боя, что были предоставлены в комплекте, было недостаточно, поэтому был использован сервис Mixamo, для создания дополнительных анимаций — магической атаки и смерти. Данный сервис использует алгоритмы привязки к костям, чтобы задать разнообразные анимации любой гуманоидной модели (рисунок 33).

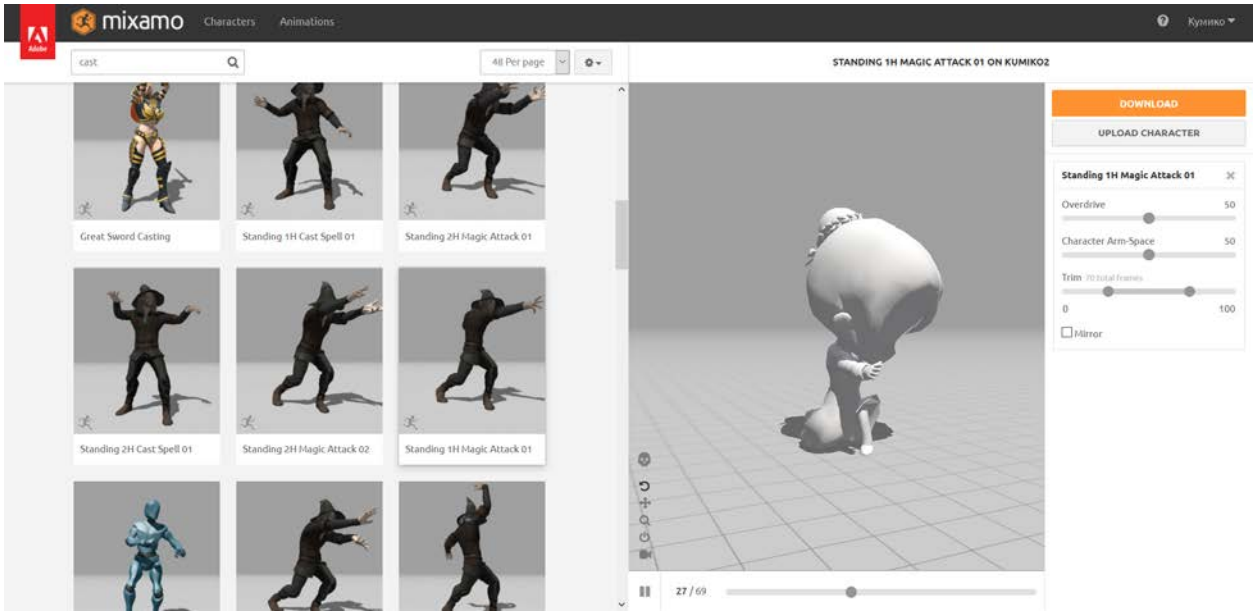

Рисунок 33 — Обработанная модель на сервисе Mixamo

После импорта всех необходимых анимаций в движок, было создано дополнительное древо анимации, отвечающее за поведение модели на поле битвы (рисунок 34).

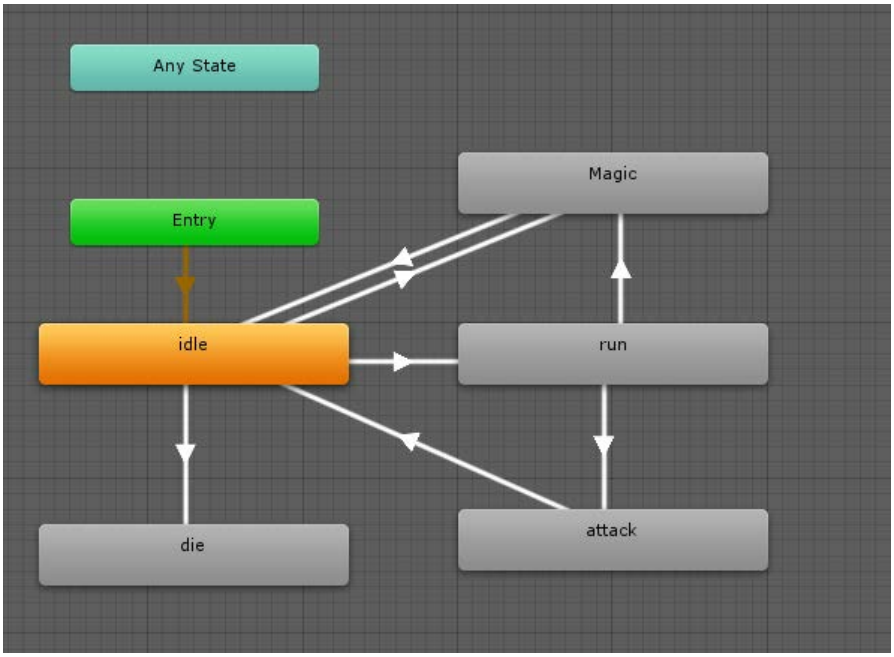

Рисунок 34 — Дерево анимаций для состояния битвы

Для моделей врагов, также были выбраны бесплатные модели монстров из магазина Unity Asset Store (рисунок 35). В них уже присутствовали все необходимые для боя анимации, и настройка дерева анимации проходила по полностью аналогичной схеме (рисунок 36).

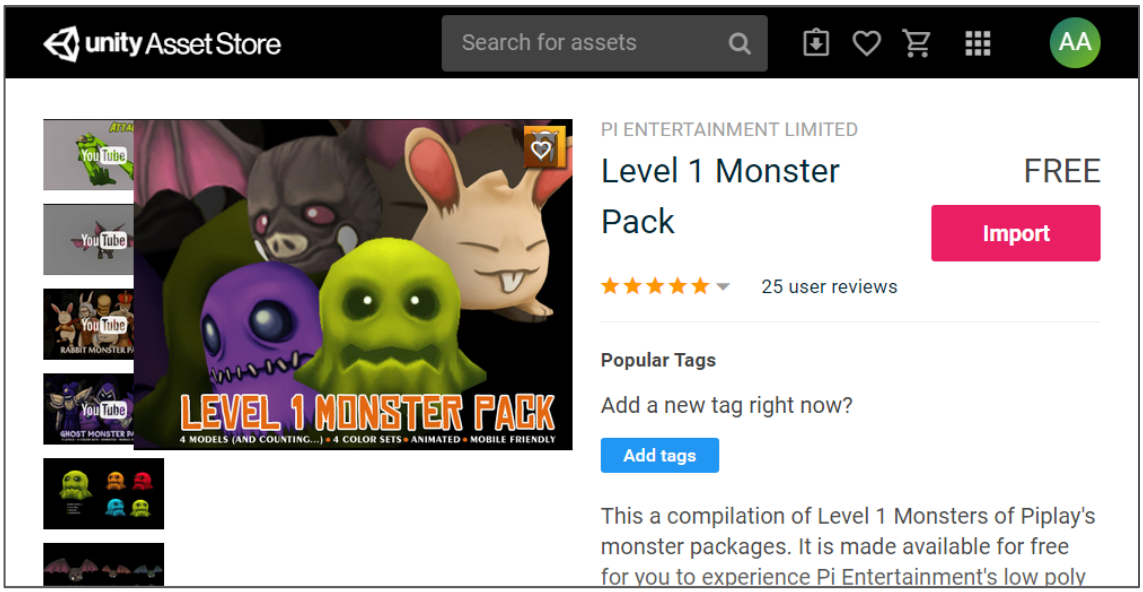

Рисунок 35 — Страница моделей врагов в Unity Asset Store

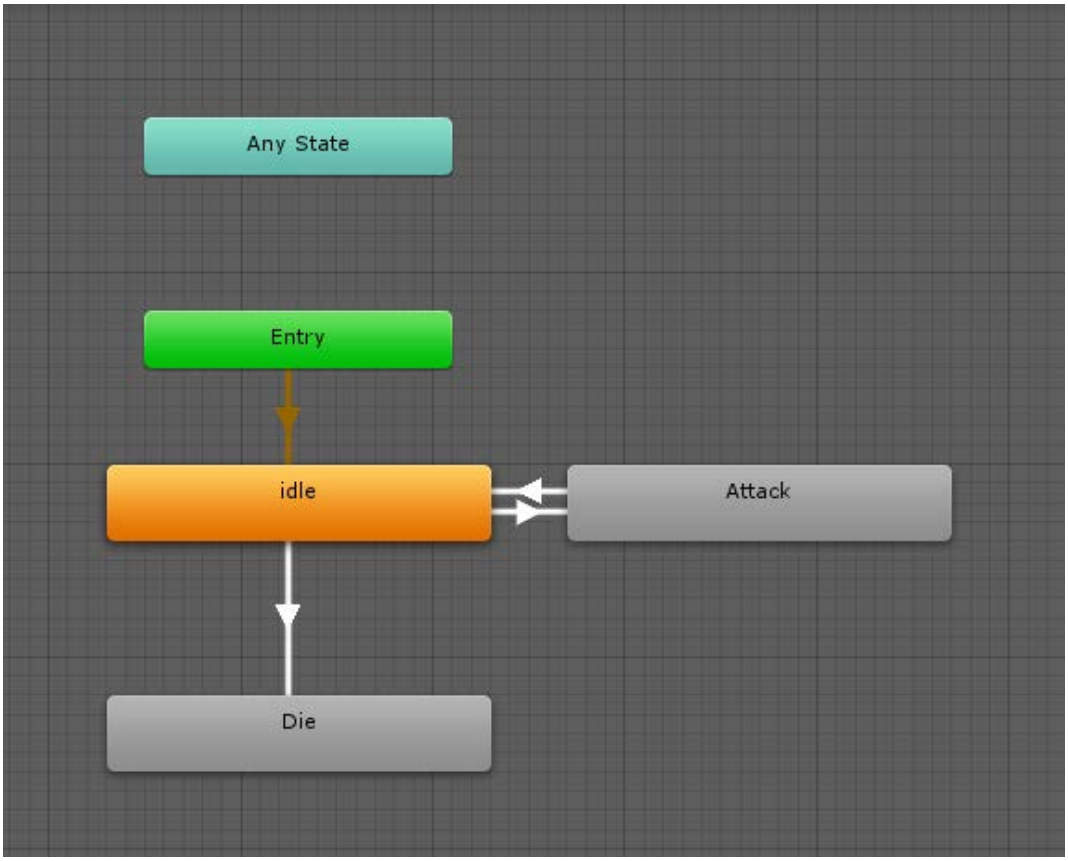

Рисунок 36 — Дерево анимаций для врага в состоянии битвы

2.3.7 Моделирование локаций

Первым этапом при создании начальной локации «Рыбацкая деревня» было создание ландшафта с помощью специальной функции Unity Engine — «Rise or Lower Terrain». В данной функции, можно детально настроить до какой высоты будет подниматься выделенная область, с какой силой и рельефом. Интерфейс функции показан на рисунке 37.

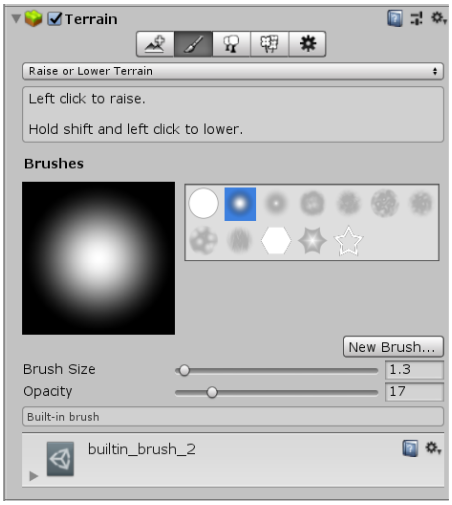

Рисунок 37 — Функция «Rise or Lower»

После создания основы ландшафта, были выбраны текстуры травы и земли с последующим созданием для них карт высот (рисунок 38). Для этого была использована функция «Paint Texture», позволяющая покрыть ландшафт определенными текстурами. Интерфейс функции показан на рисунок 39.

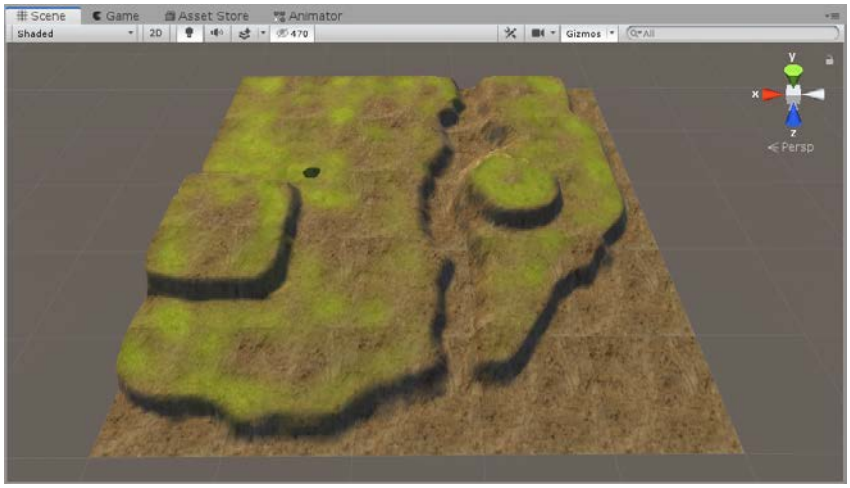

Рисунок 38 — Ландшафт с покрытием текстурами

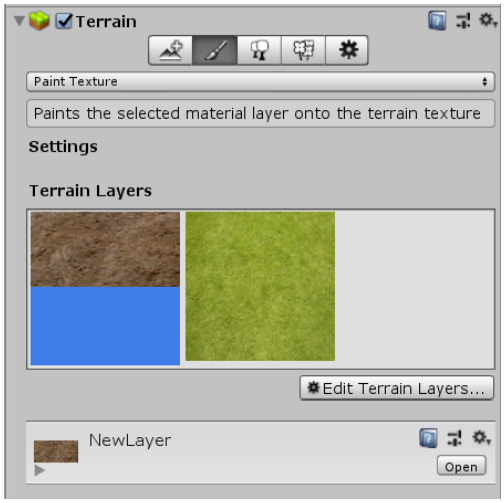

Рисунок 39 — Функция «Paint Texture»

Далее, были выбраны модели, соответствующие выбранной стилистике из бесплатного набора. Данные модели были расставлены по локации в соответствии с изначальным концептом. Также, из того же набора были выбраны деревья и камни, чтобы показать игроку границы карты (рисунок 40).

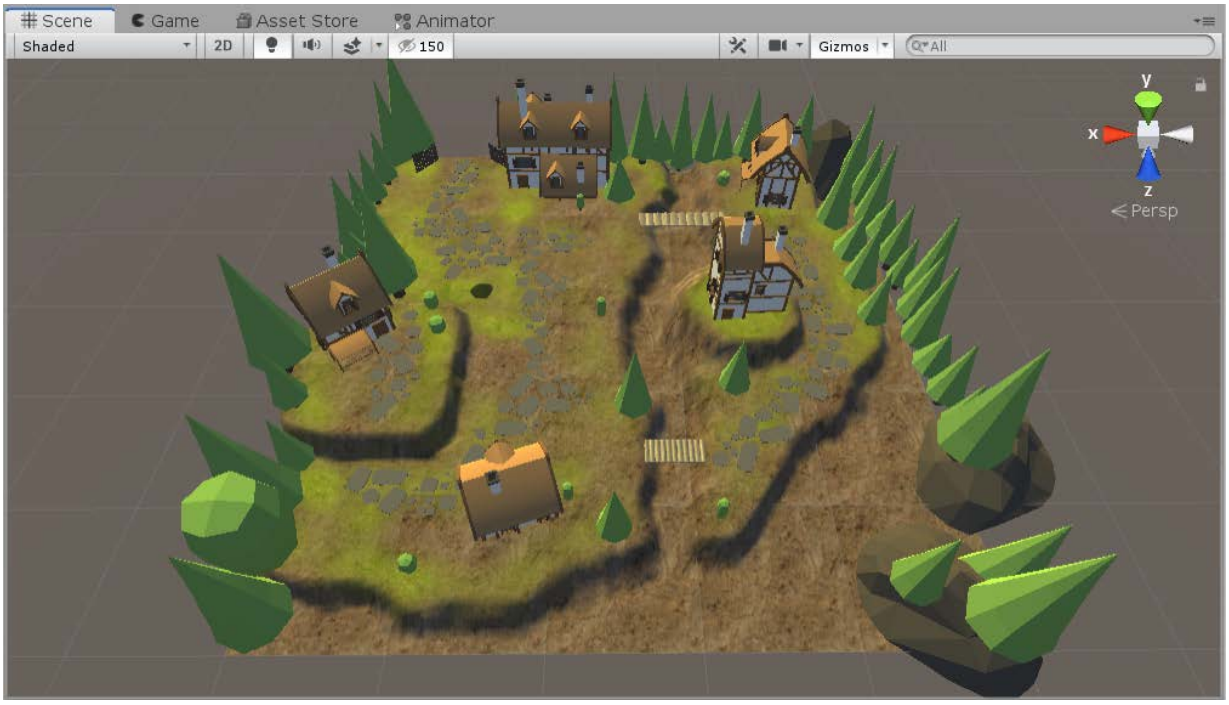

Рисунок 40 — Локация с домами и деревьями

Для создания более реалистичного вида воды, был использован бесплатный набор эффектов из Unity Asset Store. Данный набор позволяет добавить эффект волн и более реалистичного соприкосновения с другими объектами (рисунок 41).

После того как все объекты были расставлены по локации, для них были установлены невидимые границы, ограничивающие игрока, чтобы он не попал в воду или за пределы локации (рисунок 42).

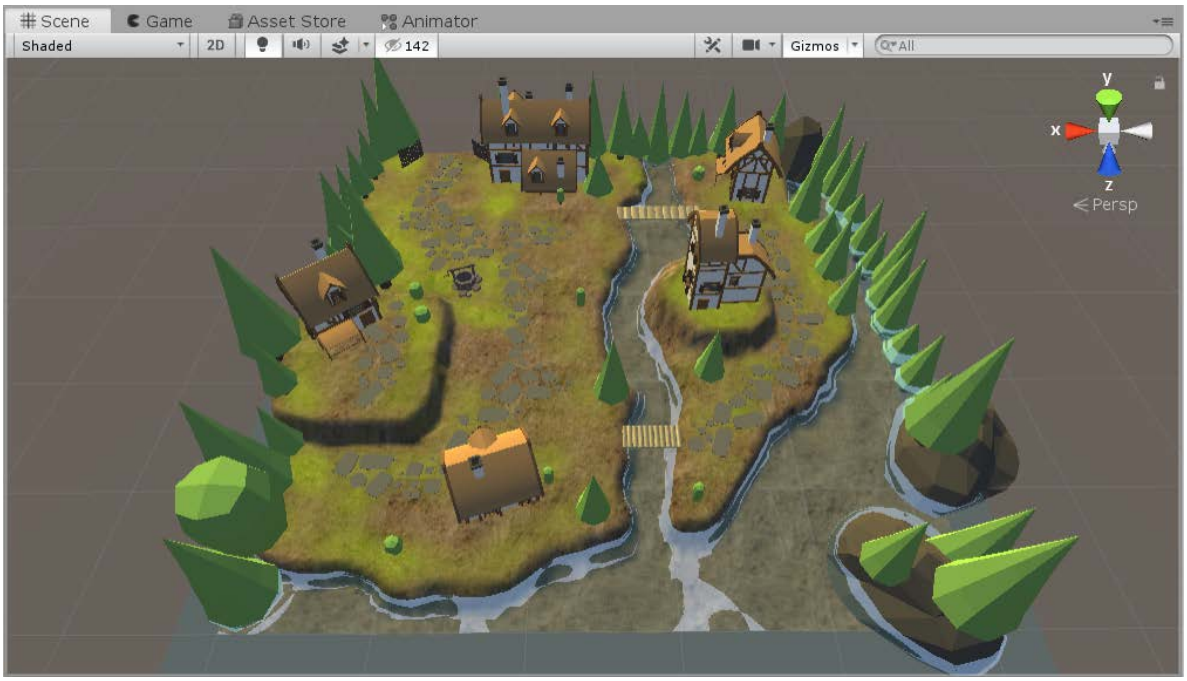

Рисунок 41 — Локация с настроенной моделью воды

Рисунок 42 — Локация с видом внутри игры

Также, было разработано три локации для домов, в которые имеется возможность зайти (рисунок 43).

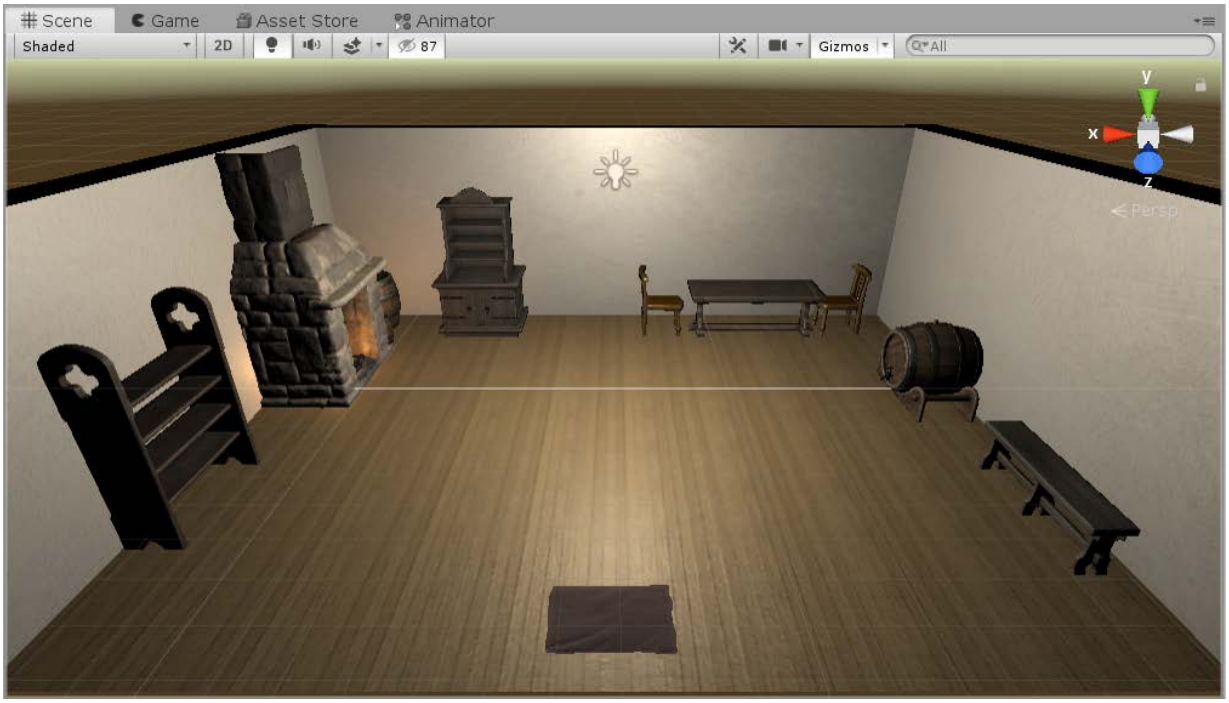

Рисунок 43 — Локация первого дома

Все модели внутри домов также были взяты из бесплатного набора моделей.

2.3.8 Тестирование мультимедийного продукта

Данный этап проходил на протяжении всего процесса разработки, так как каждая новая функция тестировалась после её добавления и при возникновении ошибок скрипты исправлялись.

Тесты помогли определиться с конечным видом боевой системы и анимаций, так как, из-за сложной структуры кода, не сменялись состояния основного автомата. Для исправления данной ошибки были упрощены анимации персонажей и добавлены кнопки возврата на предыдущий выбор.

Также, данный этап помог определиться с более удобным видом интерфейса инвентаря, так как изначальный вариант многим показался неудобным.

2.3.9 Сборка мультимедийного продукта

Финальный разработки — это сборка продукта. В Unity Engine продукт можно собрать для нескольких платформ: Android, HTML5, Windows x32 и x64, Linux, iOS и tvOS. Так как игра разрабатывалась для компьютеров и, по статистике, большинство пользователей сервиса Steam (потенциальные покупатели) используют устройства с операционной системой «Windows» с разрядностью «x64», то для сборки игры выбран именно этот формат. Перед тем как начать сборку, необходимо в настройках указать все сцены, которые будут участвовать в проекте и удалить из контента лишние папки, если таковые имеются, чтобы не увеличивать размер игры. Сборка будет проходить в фоновом режиме, по окончанию раздастся звуковой сигнал и откроется попка, в которую осуществлялась сборка.

После этого игра будет представлять из себя приложение формата *.exe и несколько вспомогательных папок (рисунок 44).

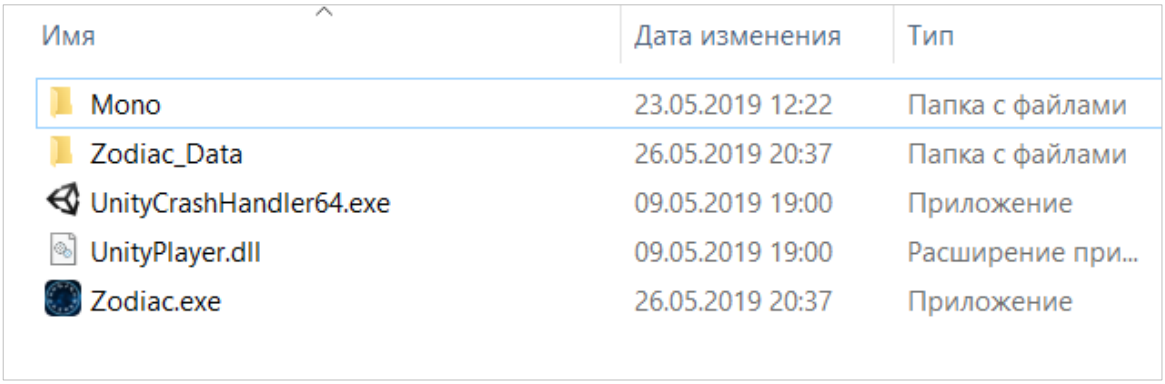

Рисунок 44 — Финал сборки игры

После сборки необходимо провести ещё несколько тестов, так как некоторые проблемы можно не обнаружить в движке. Также, после сборки игру можно переносить на устройства с выбранной операционной системой, чтобы проверить её работоспособность. На сторонних компьютерах игра запускается корректно, а значит можно её выпускать.

2.3.10 Выпуск мультимедийного продукта

На данном этапе игру повторно собирают с заполнением всей необходимой информации об игре. Добавить описание, использовать шифрование для защиты от копирования, заполнить информацию о разработчике, выбрать иконку и метод компрессии, чтобы уменьшить размер получаемой сборки (рисунок 45).

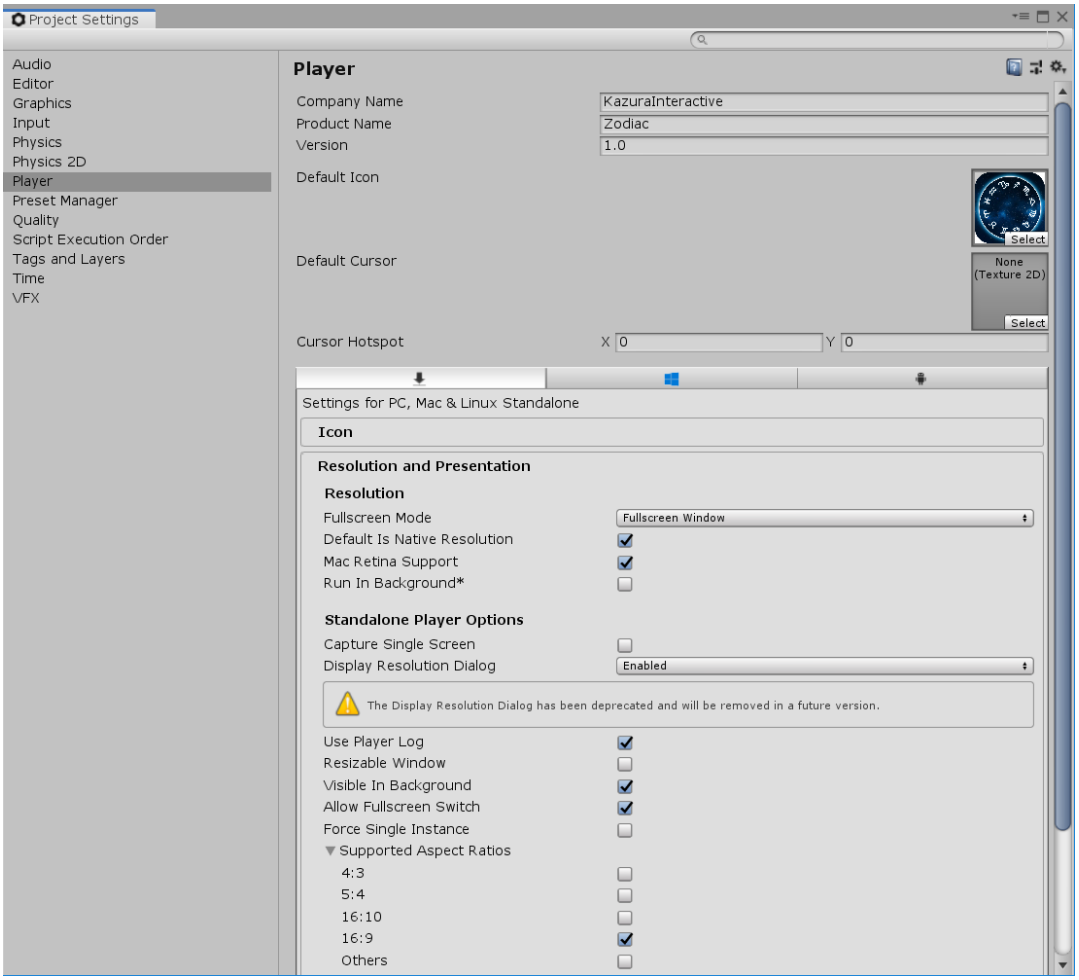

Рисунок 45 — Страница настройки информации о сборке

После финальной сборки, игру можно начать распространять.

2.4 Технические требования к мультимедийному продукту

Рекомендуемые системные требования необходимые для запуска игры указаны в таблице 4.

Таблица 4 — Рекомендуемые системные требования игры

Название	Значение
Операционная система	Windows Vista/7/8/10 (только 64-битные)
Процессор	Intel Core i7-4720HQ CPU 2.6 ГГц
Видеокарта	NVIDIA GeForce 940M
Оперативная память	8 Гб
Место на диске	2,85 Гб

2.5 Калькуляция мультимедийного продукта

В ходе выполнения выпускной квалификационной работы было использовано 5 программ: Unity Engine 2019, 3Ds Max 2019, Adobe Photoshop CC, Adobe Premier Pro.

В программе 3Ds Max проработано и сконвертировано шесть моделей.

В программе Unity Engine создано:

1. 46 материалов.
2. 47 скриптов, 6 из которых автоматы состояний.
3. 4 пользовательских интерфейса.
4. 17 объектов для настройки аудио файлов.
5. 8 сцен (6 сцен уровней, сцена главного меню, одна сцена вступительного видео).

В программе Adobe Photoshop CC обработано и создано 4 текстуры.

В игре задействовано 4 анимации врага, 5 анимаций главного героя и его сопартийца.

Всего сборок игры было 36, суммарное время сборки 4 часа.

Время, требуемое для полного прохождения всех механик игры: ~7минут.

ЗАКЛЮЧЕНИЕ

Темп развития игровой индустрии наглядно показывает актуальность тематики данной работы. Видеоигры заняли надёжную позицию в жизни современного человека будь то крупные проекты для компьютеров или небольшие головоломки для смартфонов. Благодаря доступности игровых движков и обилию информации в Интернете любой желающий может начать разрабатывать игры для компьютеров, консолей или смартфонов, что только поспособствует развитию данного сегмента рынка цифрового контента.

В рамках выпускной квалификационной работы была разработана техническая демонстрационная версия мультимедийного продукта в жанре японской ролевой пошаговой игры. Для этого было использовано специализированное программное обеспечение — игровой движок Unity Engine и его инструменты для моделирования и программирования; 3Ds Max для редактирования и обработки трехмерных моделей; Adobe Photoshop для обработки текстур; RPG Maker MV для разработки концептов персонажа и локации. Также, в работе был задействован редактор Adobe Audition для обработки звуковых эффектов и проведения конвертации в формат, который использует движок.

При работе над проектом был сделан вывод, что существует много различного софта профессионального уровня, на различных условиях, предоставляющих бесплатные лицензии. Кроме того, существует большое количество ресурсов, предоставляющих бесплатные 3D-модели, аудиофайлы и текстуры в хорошем качестве, которые можно использовать в собственных проектах. Наличие в движке Unity Engine языка программирования C#, который изучался процессе обучения в университете, позволяет упростить задачу по работе со скриптами, значительно увеличивая скорость разработки.

В соответствии с целью были выполнены задачи:

1. Изучена литература и интернет-источники по темам, связанным с жанром японских ролевых игр, а также методами разработки мультимедиа продуктов (видеоигр).

2. Проанализированы существующие мультимедиа продукты в жанре японских ролевых пошаговых игр, выявлены достоинства и недостатки.

3. Сформулированы требования к разрабатываемому мультимедиа продукту и с их учетом спроектированы дизайн.

4. Разработана архитектура мультимедийного продукта с помощью игрового движка Unity Engine.

5. Реализован и протестирован мультимедиа продукт при помощи игрового движка Unity Engine.

Таким образом, цель работы достигнута, а задачи полностью выполнены.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Ачкасова Т. А. Социально-экономическая география Японии [Текст] / Т. А. Ачкасова [и др.]. — под ред. И. С. Тихоцкая; Ассоц. японоведов, Географический фак. МГУ им. М. В. Ломоносова. — Москва: Аспект пресс, 2016. — 525 с.

2. Всемирная система оценки Metacritic [Электронный ресурс]. — Режим доступа: https://www.metacritic.com/ (дата обращения: 29.05.2019).

3. Вулф Терри Кодзима — гений. История разработчика, перевернувшего индустрию видеоигр. [Текст] / Терри Вулф. — Москва: Эксмо, 2019. — 600 с.

4. Деникин А. А. Звуковой дизайн в видеоиграх. Технологии «игрового» аудио для непрограммистов. [Текст] / А. А. Деникин. — Москва: ДМК-Пресс, 2012. — 696 с.

5. Дикинсон Крис Оптимизация игр в Unity 5. Советы и методы оптимизации приложений. [Текст] / Крис Дикинсон. — Москва: ДМК-Пресс, 2017. — 306 с.

6. Игромания [Электронный ресурс]. — Режим доступа: https://www.igromania.ru/ (дата обращения: 28.05.2019).

7. Казакова Н. Ю. Целевая аудитория гейм-дизайна и игровой процесс [Текст] / Н. Ю. Казакова, Ю. В. Назаров // Москва: Декоративное искусство и предметно-пространственная среда. Вестник МГХПА., 2015. — №1. — С. 393–414.

8. Канобу [Электронный ресурс]. — Режим доступа: https://kanobu.ru/ (дата обращения: 29.05.2019).

9. Качественный Terrain в Unity 5 [Электронный ресурс]. — Режим доступа: http://flight-dream.com/forum/index.php?topic=1921.0 (дата обращения: 29.05.2019).

10. Козырь Н. С. Индустрия видеоигр в современной отраслевой экономике [Текст] / Н. С. Козырь, А. В. Астахов // Кубань: Региональная экономика: теория и практика., 2017. — №5. — С. 953–966.

11. Компьютерные игры как искусство [Электронный ресурс]. — Режим доступа: http://gamesisart.ru/ (дата обращения: 28.05.2019).

12. Лещенко Н. Ф. История Японии [Текст]: учебник для студентов вузов / Н. Ф. Лещенко, А. Н Мещеряков, С. А. Полхов — под ред. Стрельцова Д.В. — Москва: Аспект Пресс, 2015. — 560 с.

13. Официальный YouTube канал компании Unity [Электронный ресурс]. — Режим доступа: https://www.youtube.com/channel/UCG08EqOAXJk_YXPDsAvReSg (дата обращения: 29.05.2019).

14. Официальный сайт программы RPG Maker MV [Электронный ресурс]. — Режим доступа: http://www.rpgmakerweb.com/products/programs/rpg-maker-mv (дата обращения: 29.05.2019).

15. Параззит Антуан Jrpg на ложе Прокруста. [Текст] / Антуан Параззит. — Москва: ЛитРес, 2018. — 640 с.

16. Пушакова А. Э. Япония. Введение в искусство и культуру. [Текст] / А. Э. Пушакова. — Москва: Эксмо, 2019. — 128 с.

17. Русскоязычное сообщество Unity Engine [Электронный ресурс]. — Режим доступа: http://www.unity3D.ru/distribution/index.php (дата обращения: 28.05.2019).

18. Сеть разработчиков Microsoft [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ (дата обращения: 29.05.2019).

19. Учебник по Unity 5 [Электронный ресурс]. — Режим доступа: http://websketches.ru (дата обращения: 29.05.2019).

20. Хабрахабр [Электронный ресурс]. — Режим доступа: https://habr.com/ru/ (дата обращения: 29.05.2019).

21. Шрейер Джексон Кровь, пот и пиксели. Обратная сторона индустрии видеоигр [Текст] / Джексон Шрейер. — Москва: Эксмо, 2018. — 368 с.

22. Macklin Colleen Games, Design and Play. [Text] / Colleen Macklin, John Sharp. — Boston: Addison-Wesley Professional, 2016. — 288 p.

23. Sketchfab [Электронный ресурс]. — Режим доступа: https://sketchfab.com/ (дата обращения: 29.05.2019).

24. Somberg Guy Game Audio Programming: Principles and Practices. [Text] / Guy Somberg. — CRC Press, 2016. — 312 p.

25. StopGame [Электронный ресурс]. — Режим доступа: https://stopgame.ru/ (дата обращения: 28.05.2019).

26. Unity Engine Documentation [Электронный ресурс]. — Режим доступа: https://docs.unity3D.com/ScriptReference/index.html (дата обращения: 28.05.2019).

27. Yannakakis G. N. Artificial Intelligence and Games. [Text] / G. N. Yannakakis, Julian Togelius. — London: Springer, 2018. — 337 p.

28. YouTube канал Deatrockerz [Электронный ресурс]. — Режим доступа: https://www.youtube.com/user/Deatrockerz/ (дата обращения: 29.05.2019).

29. YouTube канал xOctoManx [Электронный ресурс]. — Режим доступа: https://www.youtube.com/user/xOctoManx (дата обращения: 28.05.2019).

30. YouTube канал Гоша Дударь [Электронный ресурс]. — Режим доступа: https://www.youtube.com/channel/UCvuY904el7JvBlPbdqbfguw (дата обращения: 29.05.2019).

ПРИЛОЖЕНИЕ

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

Институт инженерно-педагогического образования

Кафедра информационных систем и технологий

Направление подготовки 09.03.02 Информационные системы и технологии

Профиль подготовки «Информационные технологии в медиаиндустрии»

УТВЕРЖДАЮ
Заведующий кафедрой

_____ И. А. Суслова
подпись и.о. фамилия

« ___ » _____ 2019 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студента (ки) *4* курса группы *ИТм-403*
Егармина Антона Александровича
фамилия, имя, отчество полностью

1. Тема ***Разработка мультимедийного продукта в жанре «японская ролевая пошаговая игра»***

утверждена распоряжением по институту от «_____» _____ 20 г. № ___

2. Руководитель ***Чернякова Татьяна Викторовна***
фамилия, имя, отчество полностью

доцент (ученая степень) *к.пед.н.* (ученое звание) *доцент кафедры ИС* (должность) *РГППУ* (место работы)

3. Место преддипломной практики *ФГАОУ РГППУ ИИПО кафедра информационных систем и технологий*

4. Исходные данные к ВКР *Преддипломная практика и источники, указанные в конце пояснительной записки*

5. Содержание текстовой части ВКР (перечень подлежащих разработке вопросов)
Изучить литературу и интернет-источники по темам
Проанализировать существующие мультимедиа продукты

Сформулировать требования и с их учетом спроектировать дизайн.
Разработать архитектуру мультимедийного продукта
Реализовать и протестировать мультимедиа продукт

6. Перечень демонстрационных материалов *презентация выполненная в MS Power Point, обзорный видеоролик по мультимедиа продукту, скомпилированное техно демо мультимедиа продукта формате исполняемого файла для ОС Windows.*

7. Календарный план выполнения выпускной квалификационной работы

№ п/п	Наименование этапа дипломной работы	Срок выполнения этапа	Процент выполнения ВКР	Отметка руководителя о выполнении
1	Сбор информации по выпускной квалификационной работе		10%	подпись
2	Выполнение работ по разрабатываемым вопросам и их изложение в пояснительной записке:		60%	подпись
2.1	Анализ существующих разработок и средств их создания		10%	подпись
2.2	Составление общего алгоритма реализации продукта		10%	подпись
2.3	Подбор начального набора примеров и источников, необходимых для создания продукта		10%	подпись
2.4	Разработка мультимедийного продукта в жанре «японская ролевая пошаговая игра»		15%	подпись
2.5	Исправление недочетов продукта и его выпуск		15%	подпись
3	Оформление текстовой части ВКР		10%	подпись
4	Выполнение демонстрационных материалов к ВКР		10%	подпись
5	Нормоконтроль		5%	подпись
6	Подготовка доклада к защите в ГЭК		5%	подпись

8. Консультанты по разделам выпускной квалификационной работы

Наименование раздела	Консультант	Задание выдал		Задание принял	
		подпись	дата	подпись	дата

Руководитель _____ _____ *Задание получил* _____ _____
подпись дата подпись студента дата

9. Дипломная работа и все материалы проанализированы.
Считаю возможным допустить ***Егармина А. А.*** к защите выпускной квалификационной работы в государственной экзаменационной комиссии.
Руководитель _____ _____
подпись дата

10. Допустить ***Егармина А. А.*** к защите выпускной квалификационной работы
фамилия и. о. студента
в государственной экзаменационной комиссии (протокол заседания кафедры от «___» _____ 20___г., № _____)
Заведующий кафедрой _____ _____
подпись дата